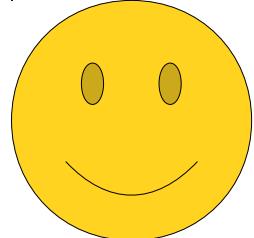


A tutorial introduction to
the QT Creator debugger

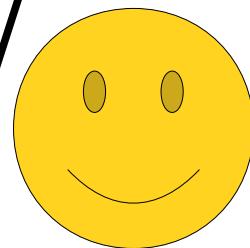
written by Keith Schwarz

Assignment 0: Using the Debugger

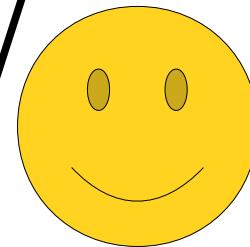


Hi everybody!

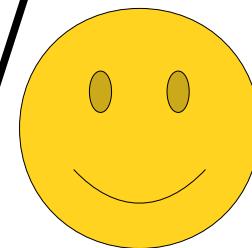
As part of Assignment 0, we'd like you to get a little bit of practice using the debugger in Qt Creator.



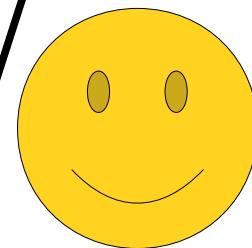
The debugger is a tool you can use to help see what your program is doing as you run it.



It's really useful for helping find errors in your programs, and the more practice you get with it, the easier it'll be to correct mistakes in the programs you write.

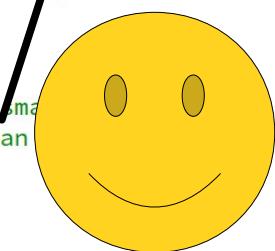


Think of this guide as a little tutorial walkthrough to help give you a sense of how to use the debugger and how to make sense of what you're seeing.



To start things off, open up the Name Hash program you ran in Part One of this assignment. Scroll down to the `nameHash` function so that you can see the entire function in your window.

```
File Edit Build Debug Analyze Tools Window Help
Projects Welcome to CS106B [master]
  -> Welcome to CS106B.pro
    -> Headers
    -> Sources
      -> lib
      -> res
      -> src
        -> NameHash.cpp
      -> Other files
NameHash.cpp
39  * th
40  * of
41  *
42  * Fo
43  * treats
44  * It then uses them as coefficients in a polynomial over
45  * F_p, where p is a large prime number, and evaluates that
46  * some smaller prime number q. (You aren't expected to know
47  * but we thought it might be fun!)
48  */
49 int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers, a large prime and a small
   * prime. These numbers were chosen because their product is less than
   * 2^31 - kLargePrime - 1.
   */
51  static const int kLargePrime = 16908799;
52  static const int kSmallPrime = 127;
53
54  int hashVal = 0;
55
56  /* Iterate across all the characters in the first name, then the last
   * name, updating the hash at each step.
   */
57  for (char ch: first + last) {
58    /* Convert the input character to lower case. The numeric values of
       * lower-case letters are always less than 127.
       */
59    ch = tolower(ch);
60    hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
61  }
62  return hashVal;
63 }
```



Sun 11:11 AM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects Welcome to CS106B [master] NameHash.cpp <Select Symbol>

Welcome to CS106B [master]

- Headers
- Sources
 - lib
 - res
 - src

Line: 31, Col: 43

39 * the meantime, think of it as a function that scrambles up the characters
40 * of the input and produces a number.
41 *
42 * For those of you who are more mathematically inclined, this function
43 * takes each character in the name and multiplies its ASCII value by a large prime
44 * number between 0 and 128.
45 * The result is then added to a running total, which is also multiplied by the same
46 * prime number. This continues until all characters have been processed.
47 * The final result is a large prime number, which is then converted back to a string.
48 *
49 * Note: The ASCII values for lower-case letters are always less than 127.
50 *
51 * The code uses two prime numbers: kSmallPrime and kLargePrime.
52 * kSmallPrime is a small prime number used to initialize the hash value.
53 * kLargePrime is a larger prime number used to prevent overflow.
54 *
55 static const int kSmallPrime =
56
57 int hashVal = 0;
58
59 /* Iterate across all the characters in the first name.
60 * name, updating the hash at each step.
61 */
62 for (char ch: first + last) {
63 /* Convert the input character to lower case. The ASCII values for
64 * lower-case letters are always less than 127.
65 */
66 ch = tolower(ch);
67 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68 }
69
70 return hashVal;
71 }

Move your mouse cursor so that it's in the space right before the line number for line 66.

Now, click the mouse!



Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

Activities Qt Creator ▾ Sun 11:12 AM NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects Welcome to CS106B [master] Sources NameHash.cpp

Welcome Sources lib res src

NameHash.cpp Other files

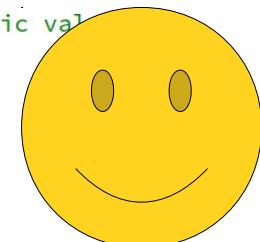
When you do, you should see a red circle with a little hourglass pop up.

This is called a **breakpoint**. If we run the program in debug mode, whenever the program gets to this line, it will pause and open up the debugger so we can see what's going on.

60 names, updating the hash at each
61 */
62 for (char ch: first + last) {
63 /* Convert the input character to lower case.
64 * lower-case letters are always less than 127.
65 */
66 ch = tolower(ch);
67 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68 }
69 return hashVal;

70 }
71 }

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results



Sun 11:12 AM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

- Welcome to CS106B [master]
 - Headers
 - Sources
 - lib
 - res
 - src
 - NameHash.cpp
- Other files

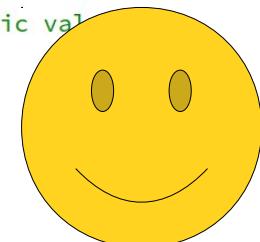
Line: 31, Col: 43

```
39 * the meantime, think of it as a function that scrambles up the characters
40 * of the input and produces a number.
41 *
42 * For those of you who are more mathematically inclined, this function
43 * treats each character in the input name as a number between 0 and 128.
44 * It then uses them as coefficients in a polynomial over the finite field
45 * F_p, where p is a large prime number, and evaluates that polynomial at
46 * some smaller prime number q. (You aren't expected to know this for CS106B,
47 * but we thought it might be fun!)
48 */
```

Now, we're going to run this program in debug mode. To do so, click on the "run in debug mode" button in the bottom-right corner of the screen. It's the one just below the regular green "run" button. When you do...

```
60     /* Name, updating the hash at each
61      */
62      for (char ch: first + last) {
63        /* Convert the input character to lower case.
64         * lower-case letters are always less than 127.
65         */
66        ch = tolower(ch);
67        hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68      }
69      return hashVal;
70    }
71 }
```

Type to locate (Ctrl...) 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 7 Test Results



... you should see something like this! Notice that a bunch of extra panels popped up in Qt Creator. We'll talk about what each of these windows mean in a second.

The screenshot shows the Qt Creator IDE interface. On the left is the Projects panel, displaying a single project named "Welcome to CS106B [master]" with a file named "NameHash.cpp" selected. The main central area is a code editor showing C++ code for a hashing function. A callout bubble points from the text "prime and a small is less than" to a yellow smiley face icon. The code editor has syntax highlighting for C++ keywords and comments. Below the code editor is the GDB debugger toolbar with buttons for stepping, breakpoints, and memory dump. The bottom status bar shows tabs for Type to locate (Ctrl...), Issues, Search Results, Application Output, Compile Output, QML Debugger Console, General Messages, and Test Results. To the right of the code editor are several floating windows: a Console window showing the prompt "What is your first name?", a Watch window titled "Col: 43" listing variables with their values and types, and a Registers window showing CPU register values. The top menu bar includes File, Edit, Build, Debug, Analyze, Tools, and Window. The title bar says "Welcome to CS106B".

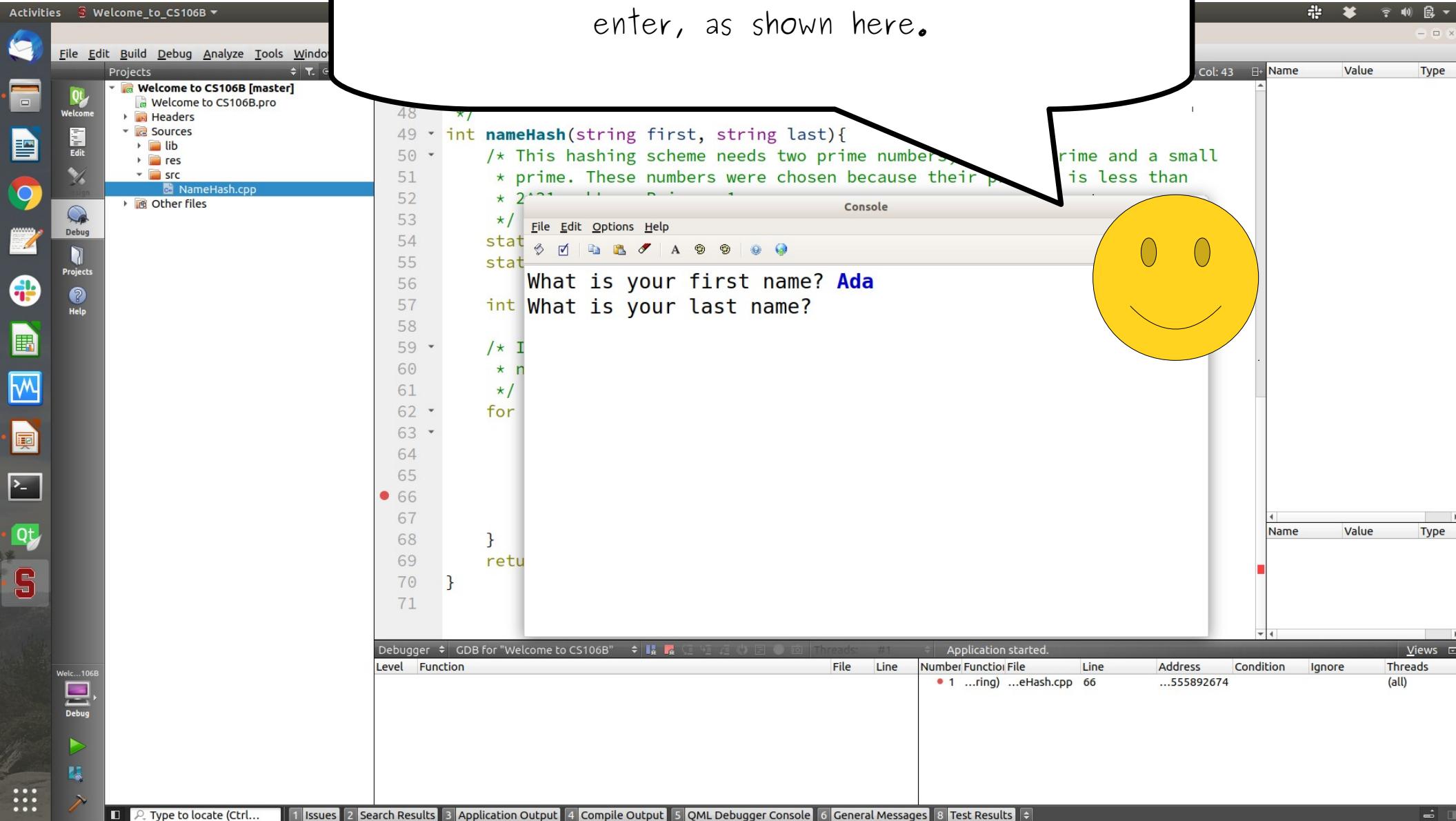
```
48     */
49     int nameHash(string first, string last){
50         /* This hashing scheme needs two prime numbers,
51         * prime. These numbers were chosen because their product
52         * 2^31 - 1 is prime.
53         */
54     stat
55     stat
56     What is your first name?
57     int
58
59     /* I
60     * n
61     */
62     for
63
64
65
66
67
68     }
69     return
70 }
71 }
```

Debugger GDB for "Welcome to CS106B" Application started.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
				1	...ring)	...eHash.cpp	66	...555892674	(all)		

Type to locate (Ctrl...) 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 7 Test Results

In the meantime, type in the first name **Ada** and hit enter, as shown here.



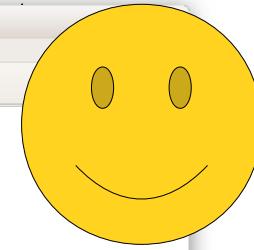
The screenshot shows the Qt Creator IDE interface. On the left is a vertical toolbar with various icons. The main area has a terminal window titled "Console". Inside the terminal, the following text is displayed:

```
What is your first name? Ada
int What is your last name?
```

A large yellow smiley face icon is overlaid on the right side of the terminal window. A callout bubble from the top-left text also points towards the smiley face. In the bottom right corner of the terminal window, there is some small, partially obscured text that appears to be part of a larger message or error stack.

Now, type in **Lovelace** as a last name, but
don't hit enter yet!

prime and a small
is less than



```
48     */
49     int nameHash(string first, string last){
50         /* This hashing scheme needs two prime numbers,
51          * prime. These numbers were chosen because their product
52          * 2^31 - 1 is prime.
53         */
54         stat
55         stat
56         What is your first name? Ada
57         int
58         What is your last name? Lovelace
59
60
61
62         for
63
64
65
66
67
68     }
69     return
70 }
71 }
```

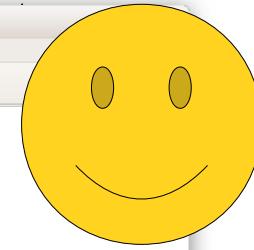
Debugger GDB for "Welcome to CS106B" Application started.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
				1	...ring)	...eHash.cpp	66	...555892674	(all)		

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

As soon as you hit enter, a bunch of things are going to pop up in Qt Creator. Don't panic! It's normal.

prime and a small
is less than



```
48 */  
49 int nameHash(string first, string last){  
50 /* This hashing scheme needs two prime numbers,  
 * prime. These numbers were chosen because their p  
 * 2^31 - 1. Both 2^31 - 1  
51 */  
52  
53  
54  
55  
56  
57  
58  
59 /* I  
60 * n  
61 */  
62 for  
63  
64  
65  
66  
67  
68 }  
69 return  
70  
71 }
```

What is your first name? Ada
int What is your last name? Lovelace

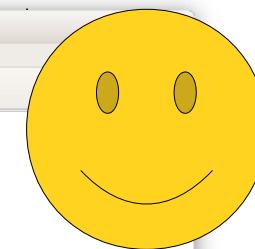
Debugger GDB for "Welcome to CS106B" Threads #1 Application started.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
				1	...ring)	...eHash.cpp	66	...555892674	(all)		

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

With that said, hit enter,
and watch the magic happen!

prime and a small
is less than



File Edit Build Debug Analyze Tools Window

Welcome to CS106B [master]

Projects

Welcome to CS106B.pro

Headers

Sources

lib

res

src

NameHash.cpp

Other files

48 */
49 int nameHash(string first, string last){
50 /* This hashing scheme needs two prime numbers,
* prime. These numbers were chosen because their product
* 2^31 - 1 is prime.
*/
51 stat
52 stat
53 What is your first name? Ada
54 int What is your last name? Lovelace
55 /* I
* n
*/
56 for
57
58
59
60
61
62
63
64
65
66
67
68 }
69 return
70
71 }

Console

File Edit Options Help

Debugger GDB for "Welcome to CS106B"

Threads #1 Application started.

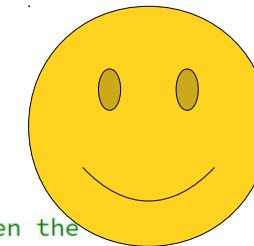
Level Function File Line Address Condition Ignore Threads

Number Function File Line Address Condition Ignore Threads

1 ...ring) ...eHash.cpp 66 ...555892674 (all)

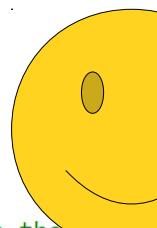
Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

Shazam! We're back in Qt Creator, and there's tons of values showing up everywhere.



prime and a small
is less than

```
48 */  
49 int nameHash(string first, string last){  
50     /* This hashing scheme needs two prime numbers.  
51      * prime. These numbers were chosen because their product  
52      * 2^31 - kLargePrime - 1.  
53      */  
54     static const int kLargePrime = 16908799;  
55     static const int kSmallPrime = 127;  
56  
57     int hashVal = 0;  
58  
59     /* Iterate across all the characters in the first name, then the  
60      * name, updating the hash at each step.  
61      */  
62     for (char ch: first + last) {  
63         /* Convert the input character to lower case. The numeric values of  
64          * lower-case letters are always less than 127.  
65          */  
66         ch = tolower(ch);  
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;  
68     }  
69     return hashVal;  
70 }
```



prime and a small prime number whose product is less than 2³¹. The two primes used are 127 and 16908799.

Name	Value
_for_begin	65 'A'
_for_end	0 '000'
_for_range	"AdaLovelace"
ch	'A'
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

Name	Value	Type
condition	Ignore	Threads

There's a lot going on right here. Let's see what's happening.

prime and a small
is less than



Name	Value
_for_begin	65 'A'
_for_end	0 '\000'
_for_range	"AdaLovelace"
ch	'A'
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

```
48     */
49     int nameHash(string first, string last){
50         /* This hashing scheme needs two prime numbers,
51          * prime. These numbers were chosen because their product
52          * 2^31 - kLargePrime - 1.
53         */
54         static const int kLargePrime = 16908799;
55         static const int kSmallPrime = 127;
56
57         int hashVal = 0;
58
59         /* Iterate across all the characters in the first name, then the
60          * name, updating the hash at each step.
61         */
62         for (char ch: first + last) {
63             /* Convert the input character to lower case. The numeric values of
64              * lower-case letters are always less than 127.
65             */
66             ch = tolower(ch);
67             hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68         }
69         return hashVal;
70     }
71 }
```

Debugger GDB for "Welcome to CS106B"
Threads: #7 Welcome_to_CS10 Stopped at breakpoint 1 in thread 7.
Level Function File Line Number Function File Line Address Condition Ignore Threads
1 nameHash Nam... 66 1 ...ring) ...eHash.cpp 66 ...555892674
2 qMain Nam... 31
3 std::function<int (), int (*)()>::_M_invoke(std::any_data const&) std_f... 302
4 std::function<int ()>::operator()() const std_f... 706
5 QtGui::lambda()::operator()(void) const spl.cpp 20981
6 std::function<int (), QtGui::startBackgroundEventLoop(GThunkin... std_f... 302
7 std::function<int ()>::operator()() const std_f... 706
8 GThreadStd::run spl.cpp 22491
9 GThreadStd::<lambda()>::operator()(void) const spl.cpp 22514
10 std::_invoke_impl<void, GThreadStd::start()::<lambda()>>(std::_invoke_o... invok... 60

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

Sun 11:18 AM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects Welcome to CS106B [master] Sources NameHash.cpp

47 * but we thought it might be fun!)
48 */
49 int nameHash(string first, string last){
50 /* This hashing scheme needs two prime numbers, a large prime and a small
51 * prime. These numbers were chosen because their product is less than
52 * $2^{31} - kLargePrime - 1$.

First, notice that our red breakpoint now has a yellow arrow in it.

the last

/* Convert the input characters to lower-case letters are added to the hash value.
 * lower-case letters are assigned numeric values from 0 to 127.
 */
ch = tolower(ch);
hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
return hashVal;

63
64
65
66
67
68
69
70
71

Debugger GDB for "Welcome to CS106B" Threads: #7 Welcome to CS106B

Level Function File Line Num
1 nameHash Nam... 66
2 qMain Nam... 31
3 std::Function_handler<int (), int (*)()>::_M_invoke(std::Any_data const&) std_f... 302
4 std::function<int ()>::operator()() const std_f... 706
5 QtGui::lambda()::operator()(void) const spl.cpp 20981
6 std::Function_handler<int (), QtGui::startBackgroundEventLoop(GThunkin... std_f... 302
7 std::function<int ()>::operator()() const std_f... 706
8 GThreadStd::run spl.cpp 22491
9 GThreadStd::<lambda()>::operator()(void) const spl.cpp 22514
10 std::_invoke_impl<void, GThreadStd::start()::<lambda()>>(std::_invoke_o... invok... 60

Name Value Type
_for_begin 65 'A'
_for_end 0 '\000'
_for_range "AdaLovelace"
ch 'A' 65
first "Ada" 0
hashVal 0
kLargePrime 16908799
kSmallPrime 127
last "Lovelace"

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

Sun 11:18 AM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

Welcome to CS106B [master]

- Welcome to CS106B.pro
- Headers
- Sources
 - lib
 - res
 - src
- NameHash.cpp
- Other files

Line: 66, Col: 9

Name	Value
_for_begin	65 'A'
_for_end	0 '\000'
_for_range	"AdaLovelace"
ch	'A'
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

This yellow arrow indicates where in the program we are right now. The program stopped running at this line because we hit that breakpoint you set earlier.

the last

```
/* Convert the input character to lower-case. The numeric values of lowercase letters are as follows:
 *   'a' = 97, 'b' = 98, ..., 'z' = 122.
 *   The numeric values of uppercase letters are as follows:
 *   'A' = 65, 'B' = 66, ..., 'Z' = 90.
 */
char ch = tolower(ch);
hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
return hashVal;
```

63 64 65 66 67 68

69 70 71

Debugger GDB for "Welcome to CS106B"

Threads: #7 Welcome to CS106B

Level	Function	File	Line	Number	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...555892674				(all)
2	qMain	Nam...	31						
3	std::function<int ()>::_M_invoke(std::any_data const&)	std_f...	302						
4	std::function<int ()>::operator()()	std_f...	706						
5	QtGui::lambda()::operator()(void)	spl.cpp	20981						
6	std::function<int ()>::operator()()	std_f...	302						
7	std::function<int ()>::operator()()	std_f...	706						
8	GThreadStd::run	spl.cpp	22491						
9	GThreadStd::<lambda()>::operator()()	spl.cpp	22514						
10	std::_invoke_impl<void, GThreadStd::start()::<lambda()>::operator()>(std::_invoke_o...	invok...	60						

Type to locate (Ctrl...) 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 7 Test Results

Sun 11:18 AM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects Welcome to CS106B [master] Sources NameHash.cpp

```

47     * but we thought it might be fun!
48     */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.

```

Name	Value
_for_begin	65 'A'
_for_end	0 '\000'
_for_range	"AdaLovelace"
ch	'A'
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

Whenever you pop up the debugger, it's good to figure out exactly where you are in the program that you're running, so you'll get into the habit of checking for this yellow arrow.

the last

/* Convert the input character to lower-case letters are added to the hash value. The numeric values of the lowercase letters are 97 through 127.

```

63 */
64     ch = tolower(ch);
65     hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
66 }
67 return hashVal;
68 }
69 }
70 }
71 }

```

Debugger GDB for "Welcome to CS106B"

Threads: #7 Welcome to CS106B

Level	Function	File	Line	Number	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...555892674				(all)
2	qMain	Nam...	31						
3	std::Function<int ()>::_M_invoke(std::Any_data const&)	std_f...	302						
4	std::function<int ()>::operator()()	std_f...	706						
5	QtGui::lambda()>::operator()(void) const	spl.cpp	20981						
6	std::Function<int ()>::operator()()	std_f...	302						
7	std::function<int ()>::operator()()	std_f...	706						
8	GThreadStd::run	spl.cpp	22491						
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514						
10	std::invoke_impl<void, GThreadStd::start()::<lambda()>::operator()>(std::function<int ()>)	invok...	60						

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

Sun 11:18 AM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - res
 - src
 - NameHash.cpp
 - Other files

Line: 66, Col: 9

Name	Value
_for_begin	65 'A'
_for_end	0 '\000'
_for_range	"AdaLovelace"
ch	'A'
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
name	"Lovelace"

47 * but we thought it might be fun!)

48 */

49 int nameHash(string first, string last){

50 /* This hashing scheme needs two prime numbers, a large prime and a small.

51 * prime. These numbers were chosen because their product is less than

52 * $2^{31} - kLargePrime - 1$.

53 */

54 static const int kLargePrime = 16908799;

55 static const int kSmallPrime = 127;

56

57 int hashVal = 0;

58

59 /* Iterate across all the characters in the first name, then the last

60 * name, updating the hash at each step.

61 */

62 for (char ch : first){

63 }

64

65

66 }

67 }

68 }

69 }

70 return hashVal;

71 }

Next, let's take a look at this panel.
This is called the call stack.

Debugger GDB for "Welcome to CS106B"

Threads: #7 Welcome_to_CS106B Stopped at breakpoint 1 in thread 7.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring)	...eHash.cpp	66	...555892674	(all)		
2	qMain	Nam...	31								
3	std::Function<int ()>::_M_invoke(std::Any_data const&)	std_f...	302								
4	std::function<int ()>::operator()()	std_f...	706								
5	QtGui::lambda()::operator()(void) const	spl.cpp	20981								
6	std::Function<int ()>::operator()()	std_f...	302								
7	std::function<int ()>::operator()()	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()()	spl.cpp	22514								
10	std::__invoke_impl<void, GThreadStd::start()::<lambda()>::operator()>(std::__invoke_o...	invok...	60								

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

Sun 11:18 AM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

Welcome to CS106B [master]

- Welcome to CS106B.pro
- Headers
- Sources
 - lib
 - res
 - src
 - NameHash.cpp
- Other files

Line: 66, Col: 9

Name	Value
_for_begin	65 'A'
_for_end	0 '\000'
_for_range	"AdaLovelace"
ch	'A'
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
name	"Lovelace"

47 * but we thought it might be fun!)

48 */

49 int nameHash(string first, string last){

50 /* This hashing scheme needs two prime numbers, a large prime and a small.

51 * prime. These numbers were chosen because their product is less than

52 * $2^{31} - kLargePrime - 1$.

53 */

54 static const int kLargePrime = 16908799;

55 static const int kSmallPrime = 127;

56

57 int hashVal = 0;

58

59 /* Iterate across all the characters in the first name, then the last

60 * name, updating the hash at each step.

61 */

62 for (char ch : first){

63 }

64 }

65 }

66 }

67 }

68 }

69 }

70 }

71 }

Right now, we know we're in the `nameHash` function, because our helpful friend the Yellow Arrow tells us exactly what line we're on!

Debugger GDB for "Welcome to CS106B"

Threads: #7 Welcome_to_CS10 ... Stopped at breakpoint 1 in thread 7.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring)	...eHash.cpp	66	...555892674	(all)		
2	qMain	Nam...	31								
3	std::Function<int ()>::_M_invoke(std::Any_data const&)	std_f...	302								
4	std::function<int ()>::operator()()	std_f...	706								
5	QtGui::lambda()::operator()(void) const	spl.cpp	20981								
6	std::Function<int ()>::operator()()	std_f...	302								
7	std::function<int ()>::operator()()	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()()	spl.cpp	22514								
10	std::invoke_impl<void, GThreadStd::start()::<lambda()>::operator()>(std::invok...	invok...	60								

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

Sun 11:18 AM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

Welcome to CS106B [master]

- Welcome to CS106B.pro
- Headers
- Sources
 - lib
 - res
 - src
- NameHash.cpp
- Other files

Line: 66, Col: 9

Name	Value
_for_begin	65 'A'
_for_end	0 '\000'
_for_range	"AdaLovelace"
ch	'A'
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
name	"Lovelace"

47 * but we thought it might be fun!)

48 */

49 int nameHash(string first, string last){

50 /* This hashing scheme needs two prime numbers, a large prime and a small.

51 * prime. These numbers were chosen because their product is less than

52 * $2^{31} - kLargePrime - 1$.

53 */

54 static const int kLargePrime = 16908799;

55 static const int kSmallPrime = 127;

56

57 int hashVal = 0;

58

59 /* Iterate across all the characters in the first name, then the last

60 * name, updating the hash at each step.

61 */

62 for (char ch : first){

63 }

64

65

66 }

67 }

68 }

69 }

70 return hashVal;

71 }

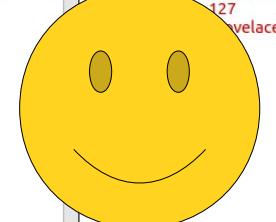
However, the yellow arrow can't tell us exactly how we got to this part of the program. What part of the program actually called `nameHash`?

Debugger GDB for "Welcome to CS106B"

Threads: #7 Welcome_to_CS106B Stopped at breakpoint 1 in thread 7.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring)	...eHash.cpp	66	...555892674	(all)		
2	qMain	Nam...	31								
3	std::Function<int ()>::_M_invoke(std::Any_data const&)	std_f...	302								
4	std::function<int ()>::operator()()	std_f...	706								
5	QGui::lambda()::operator()(void) const	spl.cpp	20981								
6	std::Function<int ()>::operator()()	std_f...	302								
7	std::function<int ()>::operator()()	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()()	spl.cpp	22514								
10	std::__invoke_impl<void, GThreadStd::start()::<lambda()>::operator()>(std::__invoke_o...	invok...	60								

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results



Sun 11:18 AM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - res
 - src
 - NameHash.cpp
 - Other files

Line: 66, Col: 9

Name	Value
_for_begin	65 'A'
_for_end	0 '\000'
_for_range	"AdaLovelace"
ch	'A'
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
name	"AdaLovelace"

47 * but we thought it might be fun!)

48 */

49 int nameHash(string first, string last){

50 /* This hashing scheme needs two prime numbers, a large prime and a small.

51 * prime. These numbers were chosen because their product is less than

52 * $2^{31} - kLargePrime - 1$.

53 */

54 static const int kLargePrime = 16908799;

55 static const int kSmallPrime = 127;

56

57 int hashVal = 0;

58

59 /* Iterate across all the characters in the first name, then the last

60 * name, updating the hash at each step.

61 */

62 for (char ch : first){

63 }

64

65

66 }

67 }

68 }

69 }

70 return hashVal;

71 }

The call stack can tell us exactly that!

Debugger GDB for "Welcome to CS106B"

Threads: #7 Welcome_to_CS106B Stopped at breakpoint 1 in thread 7.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring)	...eHash.cpp	66	...555892674	(all)		
2	qMain	Nam...	31								
3	std::Function<int ()>::_M_invoke(std::Any_data const&)	std_f...	302								
4	std::function<int ()>::operator()()	std_f...	706								
5	QGui::lambda()::operator()(void)	spl.cpp	20981								
6	std::Function<int ()>::_M_invoke(std::Any_data const&)	std_f...	302								
7	std::function<int ()>::operator()()	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void)	spl.cpp	22514								
10	std::function<void ()>::_M_invoke(std::Any_data const&)	invok...	60								

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

Sun 11:18 AM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - res
 - src
 - NameHash.cpp
 - Other files

Line: 66, Col: 9

Name	Value
_for_begin	65 'A'
_for_end	0 '\000'
_for_range	"AdaLovelace"
ch	'A'
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
name	"AdaLovelace"

47 * but we thought it might be fun!)

48 */

49 int nameHash(string first, string last){

50 /* This hashing scheme needs two prime numbers, a large prime and a small.

51 * prime. These numbers were chosen because their product is less than

52 * $2^{31} - kLargePrime - 1$.

53 */

54 static const int kLargePrime = 16908799;

55 static const int kSmallPrime = 127;

56

57 int hashVal = 0;

58

59 /* Iterate across all the characters in the first name, then the last

60 * name, updating the hash at each step.

61 */

62 for (char ch : first){

63 }

64

65

66 }

67 }

68 }

69 }

70 return hashVal;

71 }

Notice that the call stack lists a series of different functions in order. Here, it has `nameHash` (where we are now) at the top, and right below that is `qMain`.

Debugger GDB for "Welcome to CS106B"

Threads: #7 Welcome_to_CS10 ... Stopped at breakpoint 1 in thread 7.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring)	...eHash.cpp	66	...555892674	(all)		
2	qMain	Nam...	31								
3	std::Function<int ()>::_M_invoke(std::Any_data const&)	std_f...	302								
4	std::function<int ()>::operator()()	std_f...	706								
5	QGui::lambda()::operator()(void) const	spl.cpp	20981								
6	std::Function<int ()>::operator()()	std_f...	302								
7	std::function<int ()>::operator()()	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()()	spl.cpp	22514								
10	std::invoke_impl<void, GThreadStd::start()::<lambda()>::operator()>(std::invok...	invok...	60								

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

Sun 11:18 AM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - res
 - src
 - NameHash.cpp
 - Other files

Line: 66, Col: 9

Name	Value
_for_begin	65 'A'
_for_end	0 '\000'
_for_range	"AdaLovelace"
ch	'A'
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
name	"Lovelace"

47 * but we thought it might be fun!)

48 */

49 int nameHash(string first, string last){

50 /* This hashing scheme needs two prime numbers, a large prime and a small.

51 * prime. These numbers were chosen because their product is less than

52 * $2^{31} - kLargePrime - 1$.

53 */

54 static const int kLargePrime = 16908799;

55 static const int kSmallPrime = 127;

56

57 int hashVal = 0;

58

59 /* Iterate across all the characters in the first name, then the last

60 * name, updating the hash at each step.

61 */

62 for (char ch : first){

63 }

64

65

66 }

67 }

68 }

69 }

70 return hashVal;

71 }

Go and double-click the call to Main on Level 1.
When you do...

Debugger GDB for "Welcome to CS106B"

Threads: #7 Welcome_to_CS10... Stopped at breakpoint 1 in thread 7.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	namehash	Na...	66	1	...ring)	...eHash.cpp	66	...555892674	(all)		
2	qMain	Na...	31								
3	std::function<int ()>::operator()>::operator()(void) const	std_f...	302								
4	std::function<int ()>::operator()() const	std_f...	706								
5	QGui::lambda()>::operator()() const	spl.cpp	20981								
6	std::function<int ()>::operator()() const	std_f...	302								
7	std::function<int ()>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()() const	spl.cpp	22514								
10	std::invoke_<void, GThreadStd::start()::<lambda()>>(std::invok...	invok...	60								

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

Sun 11:25 AM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - res
 - src
 - NameHash.cpp
 - Other files

Line: 31, Col: 5

Name	Value	Type
first	"Ada"	std::string
hashValue	21845	int
last	"Lovelace"	std::string

#include "simpio.h" // for getLine
using namespace std;

/* Prototype for the nameHash function. This lets us use the function
 * in main and then define it later in the program.
 */
int nameHash(string first, string last);

int main() {
 string first = getLine("What is your first name? ");
 string last = getLine("What is your last name? ");

 int hashValue = nameHash(first, last);

 cout << "The hash of your name is: " << hashValue << endl;
 return 0;
}

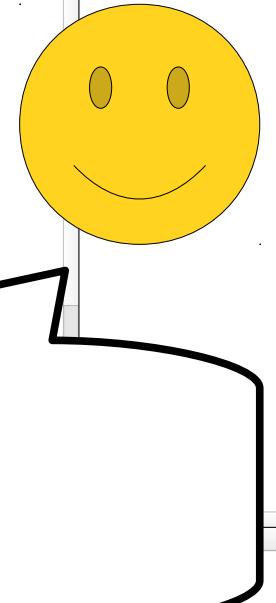
/* This
 * to ta
 * the m
 * of th
 *
 * For those
 * treats each character in the input name as a number between 0 and 128.
 * It then uses them as coefficients in a polynomial over the finite field

... you'll end up over here!

Debugger GDB for "Welcome to CS106B" Threads: #7 Welcome_to_CS10 Stopped at breakpoint 1 in thread 7.

Level	Function	File	Line
1	nameHash	Nam...	66
2	qMain	Nam...	31
3	std::function<int ()>::_M_invoke(std::Any_data const&)	std_f...	302
4	std::function<int ()>::operator()()	std_f...	706
5	QGui::lambda()::operator()(void) const	spl.cpp	20981
6	std::function<int ()>::operator()()	std_f...	302
7	std::function<int ()>::operator()()	std_f...	706
8	GThreadStd::run	spl.cpp	22491
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514
10	std::_invoke_impl<void, GThreadStd::start()::<lambda()>>(std::_invoke_o...	invok...	60

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results



Sun 11:25 AM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - res
 - src
 - NameHash.cpp
 - Other files

Line: 31, Col: 5

Name	Value	Type
first	"Ada"	std::string
hashValue	21845	int
last	"Lovelace"	std::string

#include "simpio.h" // for getLine
using namespace std;

/* Prototype for the nameHash function. This lets us use the function
 * in main and then define it later in the program.
 */
int nameHash(string first, string last);

int main() {
 string first = getLine("What is your first name? ");
 string last = getLine("What is your last name? ");

 int hashValue = nameHash(first, last);

 cout << "The hash of your name is: " << hashValue << endl;
 return 0;
}

/* This
 * to ta
 * the m
 * of th
 *
 * For those
 * treats each character in the input name as a number between 0 and 128.
 * It then uses them as coefficients in a polynomial over the finite field

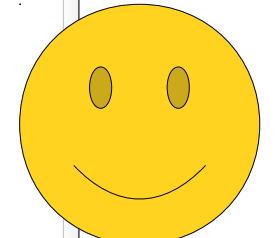
Notice that the highlighted line here includes a call to the `nameHash` function. This is the part of the code that actually called `nameHash`, which is how we got to the line with the breakpoint!

Debugger GDB for "Welcome to CS106B" Threads: #7 Welcome_to_CS10 Stopped at breakpoint 1 in thread 7.

Level	Function	File	Line
1	nameHash	Nam...	66
2	qMain	Nam...	31
3	std::function<int ()>::_M_invoke(std::Any_data const&)	std_f...	302
4	std::function<int ()>::operator()()	std_f...	706
5	QGui::lambda()::operator()(void) const	spl.cpp	20981
6	std::function<int ()>::operator()()	std_f...	302
7	std::function<int ()>::operator()()	std_f...	706
8	GThreadStd::run	spl.cpp	22491
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514
10	std::_invoke_impl<void, GThreadStd::start()::<lambda()>>(std::_invoke_o...	invok...	60

Number	Function	File	Line	Address	Condition	Ignore	Threads
1	...ring) ...eHash.cpp	66	...555892674	(all)			

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results



Sun 11:25 AM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - res
 - src
 - NameHash.cpp
 - Other files

Code Editor (NameHash.cpp)

```
19 #include "simpio.h" // for getLine
20 using namespace std;
21
22 /* Prototype for the nameHash function. This lets us use the function
   * in main and then define it later in the program.
   */
23 int nameHash(string first, string last);
24
25 int main() {
26     string first = getLine("What is your first name? ");
27     string last = getLine("What is your last name? ");
28
29     int hashValue = nameHash(first, last);
30
31     cout << "The hash of your name is: " << hashValue << endl;
32
33     return 0;
34 }
35
36 /* This
   * to ta
   * the m
   * of th
   *
   * For those
   * treats each character in the input name as a number between 0 and 128.
   * It then uses them as coefficients in a polynomial over the finite field
37 */


```

Call Stack (Threads: #7 Welcome_to_CS10)

Level	Function	File	Line
1	nameHash	Nam...	66
2	qMain	Nam...	31
3	std::function<int ()>::_M_invoke(std::any_data const&)	std_f...	302
4	std::function<int ()>::operator()()	std_f...	706
5	QGui::lambda()::operator()(void) const	spl.cpp	20981
6	std::function<int ()>::operator()()	std_f...	302
7	std::function<int ()>::operator()()	std_f...	706
8	GThreadStd::run	spl.cpp	22491
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514
10	std::_invoke_impl<void, GThreadStd::start()::<lambda()>>(std::_invoke_o...	invok...	60

Registers (Unix (LF))

Name	Value	Type
first	"Ada"	std::string
hashValue	21845	int
last	"Lovelace"	std::string

Smiley face icon

Generally speaking, you can use the call stack as a way to see which function calls got us to the point where the program paused at the breakpoint!

Sun 11:25 AM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - res
 - src
 - NameHash.cpp
 - Other files

Line: 31, Col: 5

Name	Value	Type
first	"Ada"	std::string
hashValue	21845	int
last	"Lovelace"	std::string

#include "simpio.h" // for getLine
using namespace std;

/* Prototype for the nameHash function. This lets us use the function
 * in main and then define it later in the program.
 */
int nameHash(string first, string last);

int main() {
 string first = getLine("What is your first name? ");
 string last = getLine("What is your last name? ");

 int hashValue = nameHash(first, last);

 cout << "The hash of your name is: " << hashValue << endl;
 return 0;
}

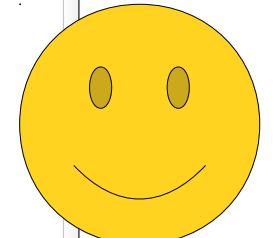
/* This
 * to ta
 * the m
 * of th
 *
 * For those
 * treats each character in the input name as a number between 0 and 128.
 * It then uses them as coefficients in a polynomial over the finite field

You might notice that there's some more stuff
in the call stack beyond just main and nameHash.
What are those?

Debugger GDB for "Welcome to CS106B" Threads: #7 Welcome_to_CS10 Stopped at breakpoint 1 in thread 7.

Level	Function	File	Line
1	nameHash	Nam...	66
2	qMain	Nam...	31
3	std::function<int ()>::_M_invoke(std::Any_data const&)	std_f...	302
4	std::function<int ()>::operator()()	std_f...	706
5	QGui::lambda()::operator()(void) const	spl.cpp	20981
6	std::function<int ()>::operator()()	std_f...	302
7	std::function<int ()>::operator()()	std_f...	706
8	GThreadStd::run	spl.cpp	22491
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514
10	std::_invoke_impl<void, GThreadStd::start()::<lambda()>>(std::_invoke_o...	invok...	60

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results



Sun 11:25 AM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - res
 - src
 - NameHash.cpp
 - Other files

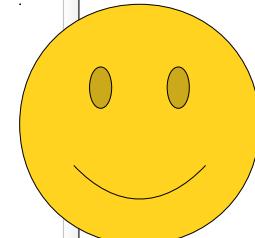
19 #include "simpio.h" // for getLine
20 using namespace std;
21
22 /* Prototype for the nameHash function. This lets us use the function
 * in main and then define it later in the program.
 */
23
24 int nameHash(string first, string last);
25
26 int main() {
27 string first = getLine("What is your first name? ");
28 string last = getLine("What is your last name? ");
29
30 int hashValue = nameHash(first, last);
31
32 cout << "The hash of your name is: " << hashValue << endl;
33 return 0;
34 }
35
36 /* This
 * to ta
 * the m
 * of th
 *
 * For those
 * treats each character in the input name as a number between 0 and 128.
 * It then uses them as coefficients in a polynomial over the finite field

Debugger GDB for "Welcome to CS106B" Threads: #7 Welcome_to_CS10 Stopped at breakpoint 1 in thread 7.

Level Function File Line Number Function File Line Address Condition Ignore Threads
1 nameHash Nam... 66 1 ...ring) ...eHash.cpp 66 ...555892674
2 qMain Nam... 31
3 std::function<int ()>::_M_invoke(std::Any_data const&) std_f... 302
4 std::function<int ()>::operator()() const std_f... 0
5 QGui::lambda()::operator()(void) const spl.cpp 20981
6 std::function<int ()>::operator()(void) const std_f... 302
7 std::function<int ()>::operator()() const std_f... 706
8 GThreadStd::run spl.cpp 22491
9 GThreadStd::<lambda()>::operator()(void) const spl.cpp 22514
10 std::function<int ()>::operator()() const std_f... 60

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

Let's find out! Double-click on the function on Level 3. (Here's what it looks like on my system; you might see something different.)



Sun 11:30 AM std_function.h - Qt Creator

Warning: The code model could not parse an included file, which might lead to incorrect code completion and highlighting, for example.

```
template<typename _Res, typename _Functor, typename... _ArgTypes>
class _Function_handler<_Res(_ArgTypes...), _Functor>
: public _Function_base::_Base_manager<_Functor>
{
    typedef _Function_base::_Base_manager<_Functor> _Base;

public:
    static _Res
    _M_invoke(const _Any_data& __functor, _ArgTypes&&... __args)
    {
        return (*_Base::_M_get_pointer(__functor))(
            std::forward<_ArgTypes>(__args)...);
    }
};

template<typename _Functor, typename... _ArgTypes>
class _Function_handler<void(_ArgTypes...), _Functor>
: public _Function_base::_Base_manager<_Functor>
{
    typedef _Function_base::_Base_manager<_Functor> _Base;

public:
    static void
    _M_invoke(const _Any_data& __functor, _ArgTypes&&... __args)
    {
        (*_Base::_M_get_pointer(__functor))(
            std::forward<_ArgTypes>(__args)...);
    }
};
```

When you do, you'll see something like this.
(This might be different depending on your OS.
Don't panic if it doesn't exactly match.)

When you do, you'll see something like this.
(This might be different depending on your OS.
Don't panic if it doesn't exactly match.)

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

Sun 11:30 AM std_function.h - Qt Creator

Warning: The code model could not parse an included file, which might lead to incorrect code completion and highlighting, for example.

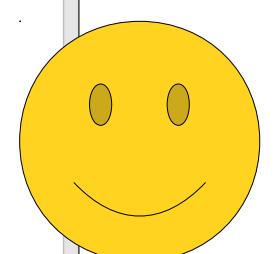
```
template<typename _Res, typename _Functor, typename... _ArgTypes>
class _Function_handler<_Res(_ArgTypes...), _Functor>
: public _Function_base::_Base_manager<_Functor>
{
    typedef _Function_base::_Base_manager<_Functor> _Base;

public:
    static _Res
    _M_invoke(const _Any_data& __functor, _ArgTypes&&... __args)
    {
        return (*_Base::_M_get_pointer(__functor))(
            std::forward<_ArgTypes>(__args)...);
    }
};

template<typename _Functor, typename... _ArgTypes>
class _Function_handler<void(_ArgTypes...), _Functor>
: public _Function_base::_Base_manager<_Functor>
{
    typedef _Function_base::_Base_manager<_Functor> _Base;

public:
    static void
    _M_invoke(const _Any_data& __functor, _ArgTypes&&... __args)
    {
        __functor(std::forward<_ArgTypes>(__args)...);
    }
};
```

Yikes! This looks Hairy and Scary! What happened?



File Edit Build Debug Analyze Tools Window Help

Projects Welcome to CS106B [master] Sources lib StanfordCPPLib spl.cpp addr2line.exe addr2line64.exe iconstrip.png silicon-large.png res SRC NameHash.cpp Other files

Debugger GDB for "Welcome...106B"

Level Function

Level	Function	Name...
1	nameHash	std_f... 302
2	qMain	std_f... 302
3	std::function<int ()>:: M_invoke(std:: Any_data const&)	std_f... 302
4	std::function<int ()>::operator()()	std_f... 706
5	QtGui::lambda()>::operator()(void) const	spl.cpp 20981
6	std::function<int ()>::operator()()	std_f... 302
7	std::function<int ()>::operator()()	std_f... 706
8	GThreadStd::run	spl.cpp 22491
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp 22514
10	std:: _invoke_<void, GThreadStd::start()::<lambda()>>(std:: _invoke_o...)	invok... 60

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

Sun 11:30 AM std_function.h - Qt Creator

Warning: The code model could not parse an included file, which might lead to incorrect code completion and highlighting, for example.

```
template<typename _Res, typename _Functor, typename... _ArgTypes>
class _Function_handler<_Res(_ArgTypes...), _Functor>
: public _Function_base::_Base_manager<_Functor>
{
    typedef _Function_base::_Base_manager<_Functor> _Base;

public:
    static _Res
    _M_invoke(const _Any_data& __functor, _ArgTypes&&... __args)
    {
        return (*_Base::_M_get_pointer(__functor))(
            std::forward<_ArgTypes>(__args)...);
    }
};

template<typename _Functor, typename... _ArgTypes>
class _Function_handler<void(_ArgTypes...), _Functor>
: public _Function_base::_Base_manager<_Functor>
{
    typedef _Function_base::_Base_manager<_Functor> _Base;

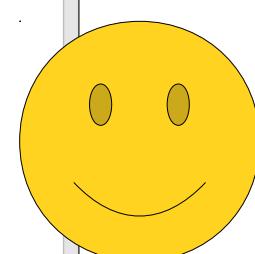
public:
    static void
    _M_invoke(const _Any_data& __functor, _ArgTypes&&... __args)
    {
        __functor(std::forward<_ArgTypes>(__args)...);
    }
};
```

Whenever you start up a program in CS106B, there's a little bit of code that we automatically call for you, which does things like setting up the console.

Debugger GDB for "Welcome to CS106B [master]"

Level	Function	Name...
1	nameHash	std_f... 302
2	qMain	std_f... 706
3	std::function<int ()>::operator()()	spl.cpp 20981
4	QtGui::lambda()::operator()()	std_f... 302
5	std::function<int ()>::operator()()	std_f... 706
6	QtGui::startBackgroundEventLoop(GThunkin...	spl.cpp 22491
7	std::function<int ()>::operator()()	std_f... 22514
8	GThreadStd::run	spl.cpp 60
9	GThreadStd::operator()()	invok... 60
10	std::invoke_<void, GThreadStd::start()::lambda()>::operator()()	

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results



Sun 11:30 AM std_function.h - Qt Creator

Warning: The code model could not parse an included file, which might lead to incorrect code completion and highlighting, for example.

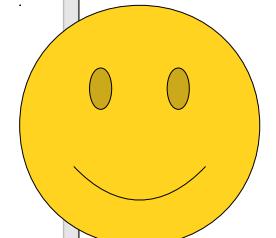
```
template<typename _Res, typename _Functor, typename... _ArgTypes>
class _Function_handler<_Res(_ArgTypes...), _Functor>
: public _Function_base::_Base_manager<_Functor>
{
    typedef _Function_base::_Base_manager<_Functor> _Base;

public:
    static _Res
    _M_invoke(const _Any_data& __functor, _ArgTypes&&... __args)
    {
        return (*_Base::_M_get_pointer(__functor))(
            std::forward<_ArgTypes>(__args)...);
    }
};

template<typename _Functor, typename... _ArgTypes>
class _Function_handler<void(_ArgTypes...), _Functor>
: public _Function_base::_Base_manager<_Functor>
{
    typedef _Function_base::_Base_manager<_Functor> _Base;

public:
    static void
    _M_invoke(const _Any_data& __functor, _ArgTypes&&... __args)
    {
        (*_Base::_M_get_pointer(__functor))(
            std::forward<_ArgTypes>(__args)...);
    }
};
```

This code will show up in the call stack below your actual program.



File Edit Build Debug Analyze Tools Window Help

Projects Welcome to CS106B [master] Sources lib StanfordCPPLib spl.cpp addr2line.exe addr2line64.exe iconstrip.png silicon-large.png res SRC NameHash.cpp Other files

Debugger GDB for "Welcome...106B"

Level	Function	Name...
1	nameHash	std_f... 302
2	qMain	std_f... 706
3	std::function<int ()>::operator()() const	spl.cpp 20981
4	QtGui::lambda()::operator()(void) const	std_f... 302
5	std::function<int ()>::operator()() const	std_f... 706
6	GThreadStd::run	spl.cpp 22491
7	GThreadStd::operator()() const	spl.cpp 22514
8	GThreadStd::start()::operator()>(std::function<void ()>)	invok... 60

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

Sun 11:30 AM std_function.h - Qt Creator

Warning: The code model could not parse an included file, which might lead to incorrect code completion and highlighting, for example.

```
template<typename _Res, typename _Functor, typename... _ArgTypes>
class _Function_handler<_Res(_ArgTypes...), _Functor>
: public _Function_base::_Base_manager<_Functor>
{
    typedef _Function_base::_Base_manager<_Functor> _Base;

public:
    static _Res
    _M_invoke(const _Any_data& __functor, _ArgTypes&&... __args)
    {
        return (*_Base::_M_get_pointer(__functor))(
            std::forward<_ArgTypes>(__args)...);
    }
};

template<typename _Functor, typename... _ArgTypes>
class _Function_handler<void(_ArgTypes...), _Functor>
: public _Function_base::_Base_manager<_Functor>
{
    typedef _Function_base::_Base_manager<_Functor> _Base;

public:
    static void
    _M_invoke(const _Any_data& __functor, _ArgTypes&&... __args)
    {
        (*_Base::_M_get_pointer(__functor))(
            std::forward<_ArgTypes>(__args)...);
    }
};
```

You shouldn't need to dig around this deep in the call stack, and if you do, it should probably be a message telling you to back up a bit back to code that you actually wrote.

File Edit Build Debug Analyze Tools Window Help

Projects Welcome to CS106B [master] Sources lib StanfordCPPLib spl.cpp addr2line.exe addr2line64.exe iconstrip.png silicon-large.png res SRC NameHash.cpp Other files

Debugger GDB for "Welcome to CS106B" Level Function 1 nameHash 2 qMain 3 std::function<int(), int(*)()>:: M_invoke(std:: Any_data const&) 4 std::function<int ()>::operator()() const 5 QtGui::lambda()>::operator()(void) const 6 std::function<int(), QtGui::startBackgroundEventLoop(GThunkin... 7 std::function<int ()>::operator()() const 8 GThreadStd::run 9 GThreadStd::<lambda()>::operator()(void) const 10 std:: _invoke_<void, GThreadStd::start()::<lambda()>>(std:: _invoke_o... Name... 31 std_f... 302 std_f... 706 spl.cpp 20981 std_f... 302 std_f... 706 spl.cpp 22491 spl.cpp 22514 invok... 60

Type to locate (Ctrl...) 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 7 Test Results

Sun 11:30 AM

std_function.h - Qt Creator

Warning: The code model could not parse an included file, which might lead to incorrect code completion and highlighting, for example.

```
template<typename _Res, typename _Functor, typename... _ArgTypes>
class _Function_handler<_Res(_ArgTypes...), _Functor>
: public _Function_base::_Base_manager<_Functor>
{
    typedef _Function_base::_Base_manager<_Functor> _Base;

public:
    static _Res
    _M_invoke(const _Any_data& __functor, _ArgTypes&&... __args)
    {
        return (*_Base::_M_get_pointer(__functor))(
            std::forward<_ArgTypes>(__args)...);
    }
};

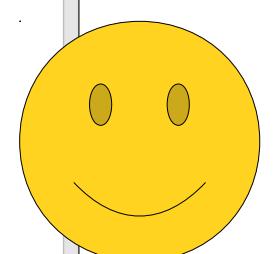
template<typename _Functor, typename... _ArgTypes>
class _Function_handler<void(_ArgTypes...), _Functor>
: public _Function_base::_Base_manager<_Functor>
{
    typedef _Function_base::_Base_manager<_Functor> _Base;

public:
    static void
    _M_invoke(const _Any_data& __functor, _ArgTypes&&... __args)
    {
        __functor(std::forward<_ArgTypes>(__args)...);
    }
};
```

Debugger GDB for "Welcome to CS106B"

Level	Function	Name...	SI
1	nameHash		
2	qMain		
3	std::function<int ()>:: M_invoke(std:: Any_data const&)	std_f... 302	
4	std::function<int ()>::operator()()	std_f... 706	
5	QtGui::lambda()>::operator()(void) const	spl.cpp 20981	
6	std::function<int ()>::operator()()	std_f... 302	
7	std::function<int ()>::operator()()	std_f... 706	
8	GThreadStd::run	spl.cpp 22491	
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp 22514	
10	std:: _invoke_<void, GThreadStd::start()::<lambda()>>(std:: _invoke_o...)	invok... 60	

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results



So let's jump back to the code that we actually wrote.

Sun 11:30 AM std_function.h - Qt Creator

Warning: The code model could not parse an included file, which might lead to incorrect code completion and highlighting, for example.

```
template<typename _Res, typename _Functor, typename... _ArgTypes>
class _Function_handler<_Res(_ArgTypes...), _Functor>
: public _Function_base::_Base_manager<_Functor>
{
    typedef _Function_base::_Base_manager<_Functor> _Base;

public:
    static _Res
    _M_invoke(const _Any_data& __functor, _ArgTypes&&... __args)
    {
        return (*_Base::_M_get_pointer(__functor))(
            std::forward<_ArgTypes>(__args)...);
    }

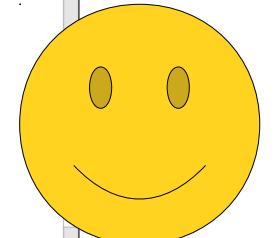
    template<class _Res, class _Functor, class... _ArgTypes>
    static _Res
    _M_invoke(const _Any_data& __functor, _ArgTypes&&... __args)
```

To do that, double-click on Level 0, the call to nameHash. When you do...

Debugger GDB for "Welcome to CS106B" Threads: #7 Welcome_to_CS10 Stopped at breakpoint 1 in thread 7.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring)	...eHash.cpp	66	...555892674	(all)		
2	QMain	Nam...	1								
3	std::function<int ()>:: M_invoke(std:: Any_data const&)	std_f...	302								
4	std::function<int ()>::operator()()	std_f...	706								
5	QtGui::lambda()>::operator()(void) const	spl.cpp	20981								
6	std::function<int ()>::operator()()	std_f...	302								
7	std::function<int ()>::operator()()	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
10	std:: invoke_ impl<void, GThreadStd::start()::<lambda()>>(std:: _invoke_o...	invok...	60								

Type to locate (Ctrl...) 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 7 Test Results



Sun 11:32 AM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

Welcome to CS106B [master]

- Welcome to CS106B.pro
- Headers
- Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
- Other files

Line: 66, Col: 9

Name	Value
_for_begin	65 'A'
_for_end	0 '\000'
_for_range	"AdaLovelace"
ch	'A' 65
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

```
47     * but we thought it might be fun!)
48     */
49     int nameHash(string first, string last){
50         /* This hashing scheme needs two prime numbers, a large prime and a small
51         * prime. These numbers were chosen because their product is less than
52         * 2^31 - kLargePrime - 1.
53         */
54         static const int kLargePrime = 16908799;
55         static const int kSmallPrime = 127;
56
57         int hashVal = 0;
58
59         /* Iterate across all the characters in the first name, then the last
60         * name, updating the hash at each step.
61         */
62         for (char ch: first + last) {
63             /* Convert the input character to lower case. The numeric values of
64             * lower-case letters are always less than 127.
65             */
66             th = tel
67         }
68     }
69
70     return hashVal;
71 }
```

Debugger GDB for "Welcome to CS106B"

Level	Function
1	nameHash
2	qMain
3	std::function<int ()>::_M_invoke(std::any_data const&)
4	std::function<int ()>::operator()()
5	QtGui::lambda()::operator()()
6	std::function<int ()>::operator()()
7	GThreadStd::run
8	GThreadStd::<lambda()>::operator()()
9	std::invoke_impl<void, GThreadStd::start()::<lambda()>>(std::function<void()>)
10	std::function<void()>::operator()()

Type to locate (Ctrl...)

Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

You'll be teleported back to safety!

Sun 11:32 AM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

Welcome to CS106B [master]

- Welcome to CS106B.pro
- Headers
- Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
- Other files

Line: 66, Col: 9

Name	Value
_for_begin	65 'A'
_for_end	0 '\000'
_for_range	"AdaLovelace"
ch	'A'
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

```
47     * but we thought it might be fun!)
48     */
49     int nameHash(string first, string last){
50         /* This hashing scheme needs two prime numbers, a large prime and a small
51         * prime. These numbers were chosen because their product is less than
52         * 2^31 - kLargePrime - 1.
53         */
54         static const int kLargePrime = 16908799;
55         static const int kSmallPrime = 127;
56
57         int hashVal = 0;
58
59         /*
60
61
62         f
63
64
65
66         hash
67     }
68
69     return hashVal;
70
71 }
```

Let's quickly recap what we've seen so far.

Debugger GDB for "Welcome to CS106B"

Threads: #7 Welcome_to_CS10 Stopped at breakpoint 1 in thread 7.

Level	Function	File	Line	Number	Function	File	Line	Address
1	nameHash	Nam...	66	1	...ring)	...eHash.cpp	66	...555892674
2	qMain	Nam...	31					
3	std::Function<int ()>::_M_invoke(std::Any_data const&)	std_f...	302					
4	std::function<int ()>::operator()()	std_f...	706					
5	QtGui::lambda()::operator()(void) const	spl.cpp	20981					
6	std::Function<int ()>::operator()()	std_f...	302					
7	std::function<int ()>::operator()()	std_f...	706					
8	GThreadStd::run	spl.cpp	22491					
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514					
10	std::_invoke_impl<void, GThreadStd::start()::<lambda()>>(std::function<int ()>)	invok...	60					

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results



Sun 11:32 AM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

Welcome to CS106B [master]

- Welcome to CS106B.pro
- Headers
- Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - silicon-large.png
 - res
 - SRC
 - NameHash.cpp
- Other files

Name Hash

Line: 66, Col: 9

```
47     * but we thought it might be fun!)
48     */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54
55     /* Convert the input character to lower case. Lower-case letters are always less than 127.
56     */
57     ch = tolower(ch);
58     hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
59 }
60 return hashVal;
61
62
63
64
65
66
67
68
69
70
71 }
```

To set a breakpoint so that we can pause the program and look around, click in the margin just before the line number where you want to pause.

values of

66

Debugger GDB for "Welcome to CS106B"

Threads: #7 Welcome_to_CS10 Stopped at breakpoint 1 in thread 7.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring)	...eHash.cpp	66	...555892674	(all)		
2	qMain	Nam...	31								
3	std::Function<int ()>::_M_invoke(std::Any_data const&)	std_f...	302								
4	std::function<int ()>::operator()()	std_f...	706								
5	QtGui::lambda()::operator()(void)	spl.cpp	20981								
6	std::Function<int ()>::operator()()	std_f...	302								
7	std::function<int ()>::operator()()	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void)	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()::<lambda()>::operator()>(std::function<int ()>)	invok...	60								

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

Sun 11:32 AM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

Welcome to CS106B [master]

- Welcome to CS106B.pro
- Headers
- Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - SRC
 - NameHash.cpp
- Other files

Name Hash

Line: 66, Col: 9

47 * but we thought it might be fun!

48 */

49 int nameHash(string first, string last){

50 /* This hashing scheme needs two prime numbers, a large prime and a small

51 * prime. These numbers were chosen because their product is less than

52 * $2^{31} - kLargePrime - 1$.

53 */

54

55

56

57

58

59

60

61

62 /* Convert the input character to lower case. Lower-case letters are always less than 127.

63 */

64

65

66 ch = tolower(ch);

67 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;

68 }

69 return hashVal;

70 }

71

Once the breakpoint is reached, it will pull up all sorts of useful information.

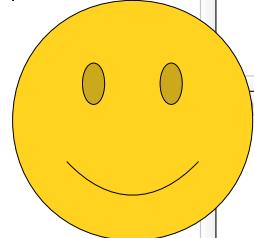
values of

Debugger GDB for "Welcome to CS106B"

Threads: #7 Welcome_to_CS10 Stopped at breakpoint 1 in thread 7.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring)	...eHash.cpp	66	...555892674	(all)		
2	qMain	Nam...	31								
3	std::function<int ()>::_M_invoke(std::any_data const&)	std_f...	302								
4	std::function<int ()>::operator()()	std_f...	706								
5	QGui::lambda()::operator()(void)	spl.cpp	20981								
6	std::function<int ()>::operator()()	std_f...	302								
7	std::function<int ()>::operator()()	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void)	spl.cpp	22514								
10	std::invoke_impl<void, GThreadStd::start()::<lambda()>::operator()>(std::invok...	invok...	60								

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results



Sun 11:32 AM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

Welcome to CS106B [master]

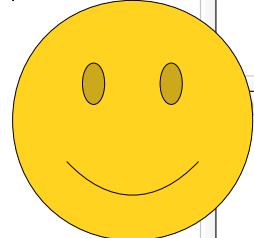
- Welcome to CS106B.pro
- Headers
- Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - SRC
 - NameHash.cpp
- Other files

Name Hash

```
47     * but we thought it might be fun!
48     */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54
55
56
57
58
59
60
61
62     /* Convert the input character to lower case. Lower-case letters are always less than 127.
63     * lower-case letters are always less than 127.
64     */
65     ch = tolower(ch);
66     hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
67 }
68
69 return hashVal;
70
71 }
```

The yellow arrow points out where we are right now.

values of



Debugger GDB for "Welcome to CS106B"

Threads: #7 Welcome_to_CS10 ... Stopped at breakpoint 1 in thread 7.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring)	...eHash.cpp	66	...555892674	(all)		
2	qMain	Nam...	31								
3	std::function<int ()>::_M_invoke(std::Any_data const&)	std_f...	302								
4	std::function<int ()>::operator()()	std_f...	706								
5	QGui::lambda()::operator()(void)	spl.cpp	20981								
6	std::function<int ()>::operator()()	std_f...	302								
7	std::function<int ()>::operator()()	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void)	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()::<lambda()>>(std::_invoke_o...	invok...	60								

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

Sun 11:32 AM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

Welcome to CS106B [master]

- Welcome to CS106B.pro
- Headers
- Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - SRC
 - NameHash.cpp
- Other files

Name Hash

Line: 66, Col: 9

```
47     * but we thought it might be fun!
48     */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54
55
56     /* Convert the input character to lower case.
57     * lower-case letters are always less than 127.
58     */
59     ch = tolower(ch);
60     hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
61 }
62
63     return hashVal;
64
65 }
```

The call stack shows us how we got into the current function.

values of

Debugger GDB for "Welcome to CS106B"

Threads: #7 Welcome_to_CS10 Stopped at breakpoint 1 in thread 7.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring)	...eHash.cpp	66	...555892674	(all)		
2	qMain	Nam...	31								
3	std::function<int ()>::_M_invoke(std::any_data const&)	std_f...	302								
4	std::function<int ()>::operator()() const	std_f...	706								
5	QGui::lambda()::operator()() const	spl.cpp	20981								
6	std::function<int ()>::operator()() const	std_f...	302								
7	std::function<int ()>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()() const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()::<lambda()>>(std::_invoke_o...	invok...	60								

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

A yellow smiley face icon is positioned next to the variable values table.

Sun 11:32 AM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

Welcome to CS106B [master]

- Welcome to CS106B.pro
- Headers
- Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - SRC
 - NameHash.cpp
- Other files

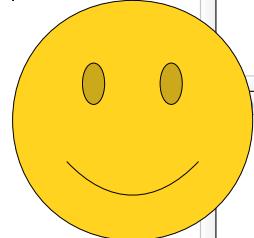
Name Hash

Line: 66, Col: 9

```
47     * but we thought it might be fun!)
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53 */
54
55     /* Convert the input character to lower case.
56     * lower-case letters are always less than 127.
57     */
58     ch = tolower(ch);
59     hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
60 }
61
62     return hashVal;
63 }
64
65
66
67
68
69
70
71 }
```

Now, let's see how we can read the values of the variables in this function.

values of



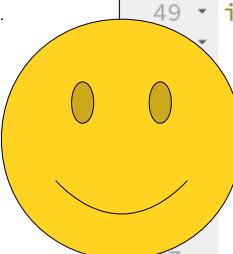
Debugger GDB for "Welcome to CS106B"

Threads: #7 Welcome_to_CS10 ... Stopped at breakpoint 1 in thread 7.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring)	...eHash.cpp	66	...555892674	(all)		
2	qMain	Nam...	31								
3	std::function<int ()>::_M_invoke(std::Any_data const&)	std_f...	302								
4	std::function<int ()>::operator()()	std_f...	706								
5	QGui::lambda()::operator()(void)	spl.cpp	20981								
6	std::function<int ()>::operator()()	std_f...	302								
7	std::function<int ()>::operator()()	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void)	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()::<lambda()>::operator()>(std::function<int ()>)	invok...	60								

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

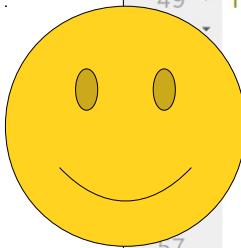
Look up at this panel over here.



```
47
48     */
49     int hash(string first, string last){
50         /* This hashing scheme needs two prime numbers, a large prime and a small
51         * prime. These numbers were chosen because their product is less than
52         * 2^31 - kLargePrime - 1.
53         */
54         static const int kLargePrime = 16908799;
55         static const int kSmallPrime = 127;
56
57         int hashVal = 0;
58
59         /* Iterate across all the characters in the first name, then the last
60         * name, updating the hash at each step.
61         */
62         for (char ch: first + last) {
63             /* Convert the input character to lower case. The numeric values of
64             * lower-case letters are always less than 127.
65             */
66             ch = tolower(ch);
67             hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68         }
69         return hashVal;
70     }
71 }
```

The screenshot shows the Qt Creator IDE interface. On the left is the project tree for "Welcome to CS106B [master]" containing "Welcome to CS106B.pro", "Headers", "Sources" (with "lib" and "src" subfolders), and "Other files". The main editor window displays the "NameHash.cpp" source code. A callout bubble points from the text "Look up at this panel over here." to the "Variables" panel on the right, which lists symbols like "_for_begin", "first", "hashVal", etc., with their values. The bottom of the screen shows the Qt Creator status bar with tabs for "Type to locate (Ctrl...)".

This window lets you take a look at all the values of the local variables that are in scope right now.



```
47
48     */
49     int
50         string first, string last){
51             /* This hashing scheme needs two prime numbers, a large prime and a small
52             * prime. These numbers were chosen because their product is less than
53             * 2^31 - kLargePrime - 1.
54             */
55             static const int kLargePrime = 16908799;
56             static const int kSmallPrime = 127;
57
58             int hashVal = 0;
59
60             /* Iterate across all the characters in the first name, then the last
61             * name, updating the hash at each step.
62             */
63             for (char ch: first + last) {
64                 /* Convert the input character to lower case. The numeric values of
65                 * lower-case letters are always less than 127.
66                 */
67                 ch = tolower(ch);
68                 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
69             }
70             return hashVal;
71 }
```

Debugger GDB for "Welcome to CS106B"

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring)	...eHash.cpp	66	...555892674	(all)		
2	qMain	Nam...	31								
3	std::function<int ()>::_M_invoke(std::any_data const&)	std_f...	302								
4	std::function<int ()>::operator()()	std_f...	706								
5	QtGui::lambda()::operator()(void) const	spl.cpp	20981								
6	std::function<int ()>::operator()()	std_f...	302								
7	std::function<int ()>::operator()()	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()::<lambda()>>(std::_invoke_o...	invok...	60								

Type to locate (Ctrl...) 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 7 Test Results

Depending on what OS you're using, these might be in a different order, and there might be some weird-looking ones in there in addition to nicer ones like ch and hashVal.

The screenshot shows the Qt Creator IDE interface. On the left is the project navigation pane with a yellow smiley face icon. The central area is the code editor displaying a C++ file named NameHash.cpp. A red circle highlights line 66, which contains the assignment `ch = tolower(ch);`. The code implements a hashing scheme for strings. The right side of the interface features a debugger window showing the call stack and registers. A large callout bubble from the top text points to the code editor near the highlighted line.

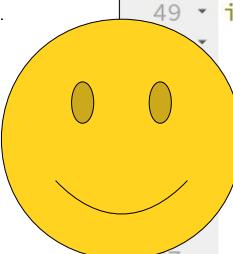
```
47
48     */
49     int
50     string first, string last){
51         /* This hashing scheme needs two prime numbers, a large prime and a small
52          * prime. These numbers were chosen because their product is less than
53          * 2^31 - kLargePrime - 1.
54         */
55         static const int kLargePrime = 16908799;
56         static const int kSmallPrime = 127;
57
58         int hashVal = 0;
59
60         /* Iterate across all the characters in the first name, then the last
61          * name, updating the hash at each step.
62         */
63         for (char ch: first + last) {
64             /* Convert the input character to lower case. The numeric values of
65              * lower-case letters are always less than 127.
66             */
67             ch = tolower(ch);
68             hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
69         }
70         return hashVal;
71     }
```

Threads: #7 Welcome_to_CS10 Stopped at breakpoint 1 in thread 7.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring)	...eHash.cpp	66	...555892674	(all)		
2	qMain	Nam...	31								
3	std::function<int ()>::_M_invoke(std::Any_data const&)	std_f...	302								
4	std::function<int ()>::operator()()	std_f...	706								
5	QGui::lambda()::operator()(void) const	spl.cpp	20981								
6	std::function<int ()>::operator()()	std_f...	302								
7	std::function<int ()>::operator()()	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()::<lambda()>>(std::_invoke_o...	invok...	60								

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

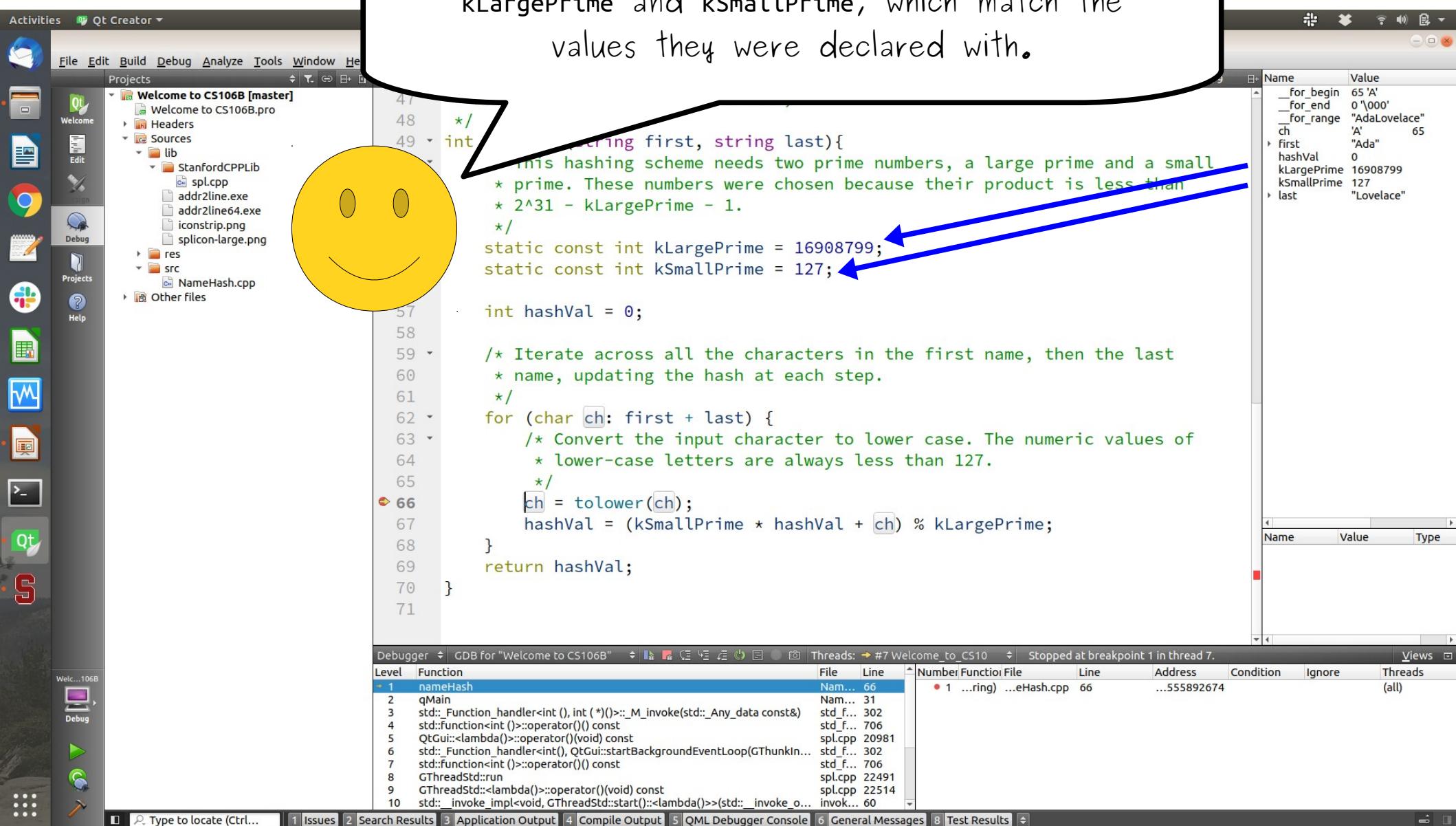
If we ignore the weird-looking ones, we can see some nice, familiar names.



```
47 */  
48  
49 int string first, string last){  
    /* This hashing scheme needs two prime numbers, a large prime and a small  
     * prime. These numbers were chosen because their product is less than  
     * 2^31 - kLargePrime - 1.  
    */  
    static const int kLargePrime = 16908799;  
    static const int kSmallPrime = 127;  
  
    int hashVal = 0;  
  
    /* Iterate across all the characters in the first name, then the last  
     * name, updating the hash at each step.  
    */  
    for (char ch: first + last) {  
        /* Convert the input character to lower case. The numeric values of  
         * lower-case letters are always less than 127.  
        */  
        ch = tolower(ch);  
        hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;  
    }  
    return hashVal;  
}
```

The screenshot shows the Qt Creator IDE interface. On the left is the project tree for "Welcome to CS106B [master]" containing "Welcome to CS106B.pro", "Headers", "Sources" (with "lib" containing "StanfordCPPLib", "spl.cpp", "addr2line.exe", "addr2line64.exe", "iconstrip.png", and "silicon-large.png"), "res", and "src" (containing "NameHash.cpp"). The main editor window displays the C++ code for "NameHash.cpp". A callout bubble from the top right points to the code, containing the handwritten note: "If we ignore the weird-looking ones, we can see some nice, familiar names." To the right of the code editor is a variable viewer showing memory dump data. At the bottom, the debugger toolbar and tabs for "Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results" are visible.

For example, here you can see the values of kLargePrime and kSmallPrime, which match the values they were declared with.



The image shows the Qt Creator IDE interface. A yellow smiley face icon is positioned on the left side of the code editor. A black callout bubble originates from the top right of the smiley face and points towards the text "which match the values they were declared with." in the main text area. The code editor displays a C++ file named NameHash.cpp. The code defines two static const integers: kLargePrime (16908799) and kSmallPrime (127). Blue arrows point from the text in the callout bubble to these two declarations. The code also includes comments explaining the hashing scheme and the choice of prime numbers. The bottom of the screen shows the Qt Creator navigation bar with tabs for Type to locate (Ctrl...), Issues, Search Results, Application Output, Compile Output, QML Debugger Console, General Messages, Test Results, and Views.

```
47 */  
48  
49 int  
50 *string first, string last){  
51     /* This hashing scheme needs two prime numbers, a large prime and a small  
52     * prime. These numbers were chosen because their product is less than  
53     * 2^31 - kLargePrime - 1.  
54     */  
55     static const int kLargePrime = 16908799;  
56     static const int kSmallPrime = 127;  
57  
58     int hashVal = 0;  
59  
60     /* Iterate across all the characters in the first name, then the last  
61     * name, updating the hash at each step.  
62     */  
63     for (char ch: first + last) {  
64         /* Convert the input character to lower case. The numeric values of  
65         * lower-case letters are always less than 127.  
66         */  
67         ch = tolower(ch);  
68         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;  
69     }  
70     return hashVal;  
71 }
```

Name	Value
_for_begin	65 'A'
_for_end	0 '\000'
_for_range	"AdaLovelace"
ch	'A' 65
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

We can also see that, at this point, hashVal is still zero.

The screenshot shows the Qt Creator IDE interface. On the left is the project navigation pane with a yellow smiley face icon. The central area is the code editor displaying a C++ file named NameHash.cpp. A blue arrow points to the line of code where the variable hashVal is initialized to 0. A callout bubble from the top right contains the handwritten note: "We can also see that, at this point, hashVal is still zero." To the right of the code editor is the debugger's registers and memory dump panes, and below is the terminal and message logs.

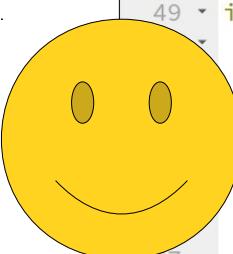
```
47     */
48
49 int
50     string first, string last){
51     /* This hashing scheme needs two prime numbers, a large prime and a small
52     * prime. These numbers were chosen because their product is less than
53     * 2^31 - kLargePrime - 1.
54     */
55     static const int kLargePrime = 16908799;
56     static const int kSmallPrime = 127,
57
58     int hashVal = 0; ←
59
60     /* Iterate across all the characters in the first name, then the last
61     * name, updating the hash at each step.
62     */
63     for (char ch: first + last) {
64         /* Convert the input character to lower case. The numeric values of
65         * lower-case letters are always less than 127.
66         */
67         ch = tolower(ch);
68         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
69     }
70     return hashVal;
71 }
```

Debugger: GDB for "Welcome to CS106B"

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring)	...eHash.cpp	66	...555892674	(all)		
2	qMain	Nam...	31								
3	std::function<int ()>::_M_invoke(std::Any_data const&)	std_f...	302								
4	std::function<int ()>::operator()()	std_f...	706								
5	QGui::lambda()::operator()(void) const	spl.cpp	20981								
6	std::function<int ()>::operator()()	std_f...	302								
7	std::function<int ()>::operator()()	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()::<lambda()>>(std::_invoke_o...	invok...	60								

Type to locate (Ctrl...) 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 7 Test Results

As we walk through the program one step at a time,
we'll see these values change.



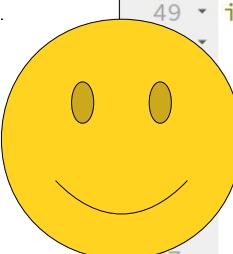
```
47 */  
48  
49 int string first, string last){  
    /* This hashing scheme needs two prime numbers, a large prime and a small  
     * prime. These numbers were chosen because their product is less than  
     * 2^31 - kLargePrime - 1.  
    */  
    static const int kLargePrime = 16908799;  
    static const int kSmallPrime = 127;  
  
    int hashVal = 0;  
  
    /* Iterate across all the characters in the first name, then the last  
     * name, updating the hash at each step.  
    */  
    for (char ch: first + last) {  
        /* Convert the input character to lower case. The numeric values of  
         * lower-case letters are always less than 127.  
        */  
        ch = tolower(ch);  
        hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;  
    }  
    return hashVal;  
}
```

Debugger GDB for "Welcome to CS106B"
Threads: #7 Welcome_to_CS10 Stopped at breakpoint 1 in thread 7.
Level Function File Line
1 nameHash Nam... 66
2 qMain Nam... 31
3 std::Function_handler<int (), int (*)()>::_M_invoke(std::Any_data const&) std_f... 302
4 std::function<int ()>::operator()() const std_f... 706
5 QtGui::lambda()::operator()(void) const spl.cpp 20981
6 std::Function_handler<int (), QtGui::startBackgroundEventLoop(GThunkin... std_f... 302
7 std::function<int ()>::operator()() const std_f... 706
8 GThreadStd::run spl.cpp 22491
9 GThreadStd::<lambda()>::operator()(void) const spl.cpp 22514
10 std::_invoke_impl<void, GThreadStd::start()::<lambda()>>(std::_invoke_o... invok... 60

Name Value
__for_begin 65 'A'
__for_end 0 '\000'
__for_range "AdaLovelace"
ch 'A' 65
first "Ada" 0
hashVal 16908799
kLargePrime 127
kSmallPrime "Lovelace"

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

Now, let's take a look at this for loop.



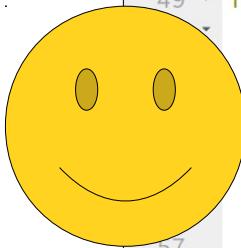
```
47 */  
48  
49 int string first, string last){  
    /* This hashing scheme needs two prime numbers, a large prime and a small  
     * prime. These numbers were chosen because their product is less than  
     * 2^31 - kLargePrime - 1.  
    */  
    static const int kLargePrime = 16908799;  
    static const int kSmallPrime = 127;  
  
    int hashVal = 0;  
  
    /* Iterate across all the characters in the first name, then the last  
     * name, updating the hash at each step.  
    */  
    for (char ch: first + last) {  
        /* Convert the input character to lower case. The numeric values of  
         * lower-case letters are always less than 127.  
        */  
        ch = tolower(ch);  
        hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;  
    }  
    return hashVal;  
}
```

The code is a C++ function named `nameHash` that takes two strings, `first` and `last`, and returns their combined hash value. It uses two prime numbers, `kLargePrime` and `kSmallPrime`, to calculate the hash. The function iterates over each character of both names, converting them to lowercase using `tolower`, and updating the hash value by multiplying it with `kSmallPrime` and adding the character's numeric value (converted to an int). A callout bubble points to the for loop, and a dashed blue rectangle highlights the loop body.

Qt Creator interface elements visible include:

- File menu: File, Edit, Build, Debug, Analyze, Tools, Window, Help
- Projects panel: Welcome to CS106B [master], Sources, Headers, lib, res, src, NameHash.cpp
- Toolbars: Standard, Qt, Help
- Bottom navigation bar: Type to locate (Ctrl...), Issues, Search Results, Application Output, Compile Output, QML Debugger Console, General Messages, Test Results
- Right-hand panels: Threads, Stopped at breakpoint 1 in thread 7, and a Variables table showing local variables.

This loop is a **range-based for loop**. It says "for each character in the string first + last, do something with that character."



```
47
48     */
49     int
50         string first, string last){
51             /* This hashing scheme needs two prime numbers, a large prime and a small
52             * prime. These numbers were chosen because their product is less than
53             * 2^31 - kLargePrime - 1.
54             */
55             static const int kLargePrime = 16908799;
56             static const int kSmallPrime = 127;
57
58             int hashVal = 0;
59
60             /* Iterate across all the characters in the first name, then the last
61             * name, updating the hash at each step.
62             */
63             for (char ch: first + last) {
64                 /* Convert the input character to lower case. The numeric values of
65                 * lower-case letters are always less than 127.
66                 */
67                 ch = tolower(ch);
68                 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
69             }
70             return hashVal;
71 }
```

The code is a C++ function named `nameHash` that takes two strings, `first` and `last`, as parameters. It initializes a variable `hashVal` to 0. Then, it iterates over each character in the concatenated string `first + last` using a range-based for loop. Inside the loop, it converts each character to lowercase using `tolower` and updates the `hashVal` by multiplying it with `kSmallPrime` and adding the character's value. Finally, it returns the calculated `hashVal`.

Breakpoint 1 is set on line 66, where the character `ch` is converted to lowercase.

Variables table:

Name	Value
_for_begin	65 'A'
_for_end	0 '\000'
_for_range	"AdaLovelace"
ch	'A' 65
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

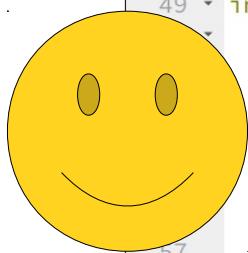
Threads table:

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring)	...eHash.cpp	66	...555892674	(all)		
2	qMain	Nam...	31								
3	std::function<int ()>::_M_invoke(std::any_data const&)	std_f...	302								
4	std::function<int ()>::operator()()	std_f...	706								
5	QGui::lambda()::operator()()	spl.cpp	20981								
6	std::function<int ()>::operator()()	std_f...	302								
7	std::function<int ()>::operator()()	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()()	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()::<lambda()>>(std::_invoke_o...	invok...	60								

Bottom navigation bar:

- Type to locate (Ctrl...)
- Issues
- Search Results
- Application Output
- Compile Output
- QML Debugger Console
- General Messages
- Test Results

Remember (from a while back) that we entered the name Ada Lovelace.



```
    string first, string last){  
    This hashing scheme needs two prime numbers, a large prime and a small  
    * prime. These numbers were chosen because their product is less than  
    * 2^31 - kLargePrime - 1.  
    */  
    static const int kLargePrime = 16908799;  
    static const int kSmallPrime = 127;  
  
    int hashVal = 0;  
  
    /* Iterate across all the characters in the first name, then the last  
     * name, updating the hash at each step.  
    */  
    for (char ch: first + last) {  
        /* Convert the input character to lower case. The numeric values of  
         * lower-case letters are always less than 127.  
        */  
        ch = tolower(ch);  
        hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;  
    }  
    return hashVal;
```

Name	Value
<code>_for_begin</code>	65 'A'
<code>_for_end</code>	0 '000'
<code>_for_range</code>	"AdaLovelace"
<code>ch</code>	'A'
<code>first</code>	"Ada"
<code>hashVal</code>	0
<code>kLargePrime</code>	16908799
<code>kSmallPrime</code>	127
<code>last</code>	"Lovelace"

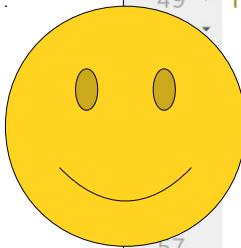
Name	Value	Type
------	-------	------

Condition	Ignore	Threads (<small>2/10</small>)

Level	Function	File	Line
- 1	nameHash	Nam...	66
2	qMain	Nam...	31
3	std:: Function_handler<int ()>, int (*)()>::_M_invoke(std:: Any_data const&)	std f...	302
4	std::function<int ()>::operator()() const	std f...	706
5	QtGui::<lambda()>::operator()(void) const	spl.cpp	20981
6	std:: Function_handler<int (), QtGui::startBackgroundEventLoop(GThunkInfo...>::operator()() const	std f...	302
7	std::function<int ()>::operator()() const	std f...	706
8	GThreadStd::run	spl.cpp	22491
9	GThreadStd:<lambda()>::operator()(void) const	spl.cpp	22514
10	std:: invoke impl<void (GThreadStd::start()):<lambda()>>(std:: invoke o...	invok...	60

Income_to_CS10	Stopped at breakpoint 1 in thread 7.			
Number	Function	File	Line	Address
● 1	...ring)	...eHash.cpp	66	...555892674

If we take a look at the current value of the variable `ch`, we can see that it has the value A. That's the first letter of the name Ada Lovelace.



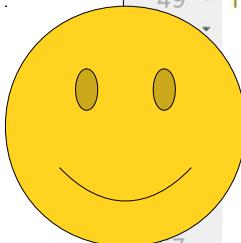
```
File Edit Build Debug Analyze Tools Window Help
Projects
  Welcome to CS106B [master]
    Welcome to CS106B.pro
    Headers
    Sources
      lib
        StanfordCPPLib
          spl.cpp
          addr2line.exe
          addr2line64.exe
          iconstrip.png
          splicon-large.png
      res
      src
        NameHash.cpp
    Other files
Edit
Design
Debug
Projects
Help
Activities Qt Creator
Name Value
__for_begin 65 'A'
__for_end 0 '\000'
__for_range "AdaLovelace"
ch A' 65
First Ada
hashVal 0
kLargePrime 16908799
kSmallPrime 127
last "Lovelace"
Name Value Type
Name HashVal
Threads: #7 Welcome_to_CS10
Stopped at breakpoint 1 in thread 7.
Level Function File Line
1 nameHash Nam... 66
2 qMain Nam... 31
3 std::Function_handler<int (), int (*)()>::_M_invoke(std::Any_data const&) std_f... 302
4 std::function<int ()>::operator()() const std_f... 706
5 QtGui::lambda()::operator()(void) const spl.cpp 20981
6 std::Function_handler<int (), QtGui::startBackgroundEventLoop(GThunkin... std_f... 302
7 std::function<int ()>::operator()() const std_f... 706
8 GThreadStd::run spl.cpp 22491
9 GThreadStd::<lambda()>::operator()(void) const spl.cpp 22514
10 std::_invoke_impl<void, GThreadStd::start()::<lambda()>>(std::_invoke_o... invok... 60
Number Function File Line Address Condition Ignore Threads
1 ...ring) ...eHash.cpp 66 ...555892674 (all)
Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results
```

47 */
48
49 int string first, string last){
 /* This hashing scheme needs two prime numbers, a large prime and a small
 * prime. These numbers were chosen because their product is less than
 * $2^{31} - kLargePrime - 1$.
 */
 static const int kLargePrime = 16908799;
 static const int kSmallPrime = 127;

 int hashVal = 0;

 /* Iterate across all the characters in the first name, then the last
 * name, updating the hash at each step.
 */
 for (char ch: first + last) {
 /* Convert the input character to lower case. The numeric values of
 * lower-case letters are always less than 127.
 */
 ch = tolower(ch);
 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
 }
 return hashVal;
}

So now we know where we are (line 66), how we got there (main called nameHash), and the values in the program at this point.



```
File Edit Build Debug Analyze Tools Window Help
Projects
  Welcome to CS106B [master]
    Welcome to CS106B.pro
    Headers
    Sources
      lib
        StanfordCPPLib
          spl.cpp
          addr2line.exe
          addr2line64.exe
          iconstrip.png
          splicon-large.png
      res
      src
        NameHash.cpp
    Other files
  Activities Qt Creator
  Activities
  Activities
```

47
48
49 int
 /* string first, string last){
 This hashing scheme needs two prime numbers, a large prime and a small
 * prime. These numbers were chosen because their product is less than
 * $2^{31} - kLargePrime - 1$.
 */
static const int kLargePrime = 16908799;
static const int kSmallPrime = 127;

int hashVal = 0;

/* Iterate across all the characters in the first name, then the last
 * name, updating the hash at each step.
*/
for (char ch: first + last) {
 /* Convert the input character to lower case. The numeric values of
 * lower-case letters are always less than 127.
 */
 ch = tolower(ch);
 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
}
return hashVal;

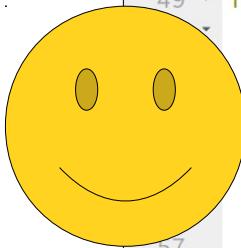
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71

Debugger GDB for "Welcome to CS106B" Threads: #7 Welcome_to_CS10 Stopped at breakpoint 1 in thread 7.
Level Function File Line
1 nameHash Nam... 66
2 qMain Nam... 31
3 std::function<int ()>::_M_invoke(std::any_data const&) std_f... 302
4 std::function<int ()>::operator()() const std_f... 706
5 QtGui::lambda()::operator()(void) const spl.cpp 20981
6 std::function<int ()>::operator()() const std_f... 302
7 std::function<int ()>::operator()() const std_f... 706
8 GThreadStd::run spl.cpp 22491
9 GThreadStd::<lambda()>::operator()(void) const spl.cpp 22514
10 std::__invoke_impl<void, GThreadStd::start()::<lambda()>>(std::__invoke_o... invok... 60

Name Value
__for_begin 65 'A'
__for_end 0 '\000'
__for_range "AdaLovelace"
ch 'A' 65
first "Ada" 0
hashVal 0
kLargePrime 16908799
kSmallPrime 127
last "Lovelace"

Type to locate (Ctrl...) 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 7 Test Results

Now, let's do something really cool - we're going to run this program one line at a time, watching what happens at each step!



```
47 */  
48  
49 int  
    string first, string last){  
    /* This hashing scheme needs two prime numbers, a large prime and a small  
     * prime. These numbers were chosen because their product is less than  
     * 2^31 - kLargePrime - 1.  
     */  
    static const int kLargePrime = 16908799;  
    static const int kSmallPrime = 127;  
  
    int hashVal = 0;  
  
    /* Iterate across all the characters in the first name, then the last  
     * name, updating the hash at each step.  
     */  
    for (char ch: first + last) {  
        /* Convert the input character to lower case. The numeric values of  
         * lower-case letters are always less than 127.  
         */  
        ch = tolower(ch);  
        hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;  
    }  
    return hashVal;  
}
```

The code is a C++ function named `nameHash` that takes two strings, `first` and `last`, and returns an integer hash value. It uses two prime numbers, `kLargePrime` and `kSmallPrime`, to calculate the hash. The function iterates over each character of both names, converting them to lowercase using `tolower`, and updates the hash value using modular arithmetic.

Breakpoint 1 is set on line 66, where the character `ch` is being converted to lowercase. The debugger shows the current state of variables:

Name	Value
<code>_for_begin</code>	65 'A'
<code>_for_end</code>	0 '\000'
<code>_for_range</code>	"AdaLovelace"
<code>ch</code>	'A' 65
<code>first</code>	"Ada"
<code>hashVal</code>	0
<code>kLargePrime</code>	16908799
<code>kSmallPrime</code>	127
<code>last</code>	"Lovelace"

The debugger also shows the call stack and threads:

Level	Function	File	Line
1	<code>nameHash</code>	Nam...	66
2	<code>qMain</code>	Nam...	31
3	<code>std::Function_handler<int (), int (*)()>::_M_invoke(std::Any_data const&)</code>	std_f...	302
4	<code>std::function<int ()>::operator()()</code>	std_f...	706
5	<code>QtGui::lambda()::operator()()</code>	spl.cpp	20981
6	<code>std::Function_handler<int (), QtGui::startBackgroundEventLoop(GThunkin...</code>	std_f...	302
7	<code>std::function<int ()>::operator()()</code>	std_f...	706
8	<code>GThreadStd::run</code>	spl.cpp	22491
9	<code>GThreadStd::<lambda()>::operator()()</code>	spl.cpp	22514
10	<code>std::_invoke_impl<void, GThreadStd::start()::<lambda()>>(std::_invoke_o...</code>	invok...	60

Threads: #7 Welcome_to_CS10 Stopped at breakpoint 1 in thread 7.

Views: (all)

Sun 11:32 AM
NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

Welcome to CS106B [master]

- Welcome to CS106B.pro
- Headers
- Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - silicon-large.png
 - res
 - src
 - NameHash.cpp
- Other files

Name Hash

```
47     * but we thought it might be fun!
48     */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the last
60     * name, updating the hash at each step.
61     */
62     for (char c : first)
63     {
64         hashVal = (hashVal * kSmallPrime + c) % kLargePrime;
65     }
66     for (char c : last)
67     {
68         hashVal = (hashVal * kSmallPrime + c) % kLargePrime;
69     }
70
71 }
```

Right above the stack trace, you'll see there are some small button icons.

Debugger GDB for "Welcome to CS106B"

Threads: #7 Welcome_to_CS10 Stopped at breakpoint 1 in thread 7.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring)	...eHash.cpp	66	...555892674	(all)		
2	qMain	Nam...	31								
3	std::Function<int ()>::_M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int ()>::operator()()	std_f...	706								
5	QGui::lambda()::operator()(void)	spl.cpp	20981								
6	std::Function<int ()>::operator()()	std_f...	302								
7	std::function<int ()>::operator()()	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void)	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()::<lambda()>>(std::_invoke_o...	invok...	60								

Type to locate (Ctrl...) 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 7 Test Results

Sun 11:32 AM
NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

Line: 66, Col: 9

Name	Value
_for_begin	65 'A'
_for_end	0 '\000'
_for_range	"AdaLovelace"
ch	'A'
first	"Ada"
hashVal	0
kLargePrime	16908799
ks	"

47 * but we thought it might be fun!)

48 */

49 int nameHash(string first, string last){

50 /* This hashing scheme needs two prime numbers, a large prime and a small

51 * prime. These numbers were chosen because their product is less than

52 * $2^{31} - kLargePrime - 1$.

53 */

54 static const int kLargePrime = 16908799;

55 static const int kSmallPrime = 127;

56

57 int hashVal = 0;

58

59 /* Iterate across all the characters in the first name, then the last

60 * name, updating the hash at each step.

61 */

62 for (char c : first){

63 }

64

65

66 }
67 }

68 }

69 return hashVal;

70 }

71 }

These buttons let you resume the program, stop the program, walk through it one line at a time, etc.

Debugger GDB for "Welcome to CS106B" (Run/Stop Buttons) Threads: #7 Welcome_to_CS10 ... Stopped at breakpoint 1 in thread 7.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring)	...eHash.cpp	66	...555892674	(all)		
2	qMain	Nam...	31								
3	std::function<int ()>::_M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int ()>::operator()()	std_f...	706								
5	QGui::lambda()::operator()(void)	spl.cpp	20981								
6	std::function<int ()>::operator()()	std_f...	302								
7	std::function<int ()>::operator()()	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void)	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()::<lambda()>>(std::_invoke_o...	invok...	60								

Type to locate (Ctrl...) 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 7 Test Results

Sun 11:32 AM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

Welcome to CS106B [master]

- Welcome to CS106B.pro
- Headers
- Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
- Other files

Line: 66, Col: 9

Name	Value
_for_begin	65 'A'
_for_end	0 '\000'
_for_range	"AdaLovelace"
ch	'A' 65
first	"Ada"
hashVal	0
kLargePrime	16908799
ks	"

47 * but we thought it might be fun!)

48 */

49 int nameHash(string first, string last){

50 /* This hashing scheme needs two prime numbers, a large prime and a small

51 * prime. These numbers were chosen because their product is less than

52 * $2^{31} - kLargePrime - 1$.

53 */

54 static const int kLargePrime = 16908799;

55 static const int kSmallPrime = 127;

56

57 int hashVal = 0;

58

59 /* Iterate across all the characters in the first name, then the last

60 * name, updating the hash at each step.

61 */

62 for (char c : first){

63 }

64

65

66 }

67

68 }

69 }

70 return hashVal;

71 }

Move your mouse so that you're hovering over the button that's third from the left. If you hover over it, it should say "step over."

Debugger GDB for "Welcome to CS106B"  Threads: #7 Welcome_to_CS10 ... Stopped at breakpoint 1 in thread 7.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring)	...eHash.cpp	66	...555892674	(all)		
2	qMain	Nam...	31								
3	std::Function<int ()>::_M_invoke(std::Any_data const&)	std_f...	302								
4	std::function<int ()>::operator()()	std_f...	706								
5	QGui::lambda()::operator()(void)	spl.cpp	20981								
6	std::Function<int ()>::operator(), QtGui::startBackgroundEventLoop(GThunkin...	std_f...	302								
7	std::function<int ()>::operator()()	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void)	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()::<lambda()>>(std::_invoke_o...	invok...	60								

Type to locate (Ctrl...) 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 7 Test Results

Sun 11:32 AM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

Welcome to CS106B [master]

- Welcome to CS106B.pro
- Headers
- Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
- Other files

Line: 66, Col: 9

Name Value

_for_begin	65	'A'
_for_end	0	'\000'
_for_range	"AdaLovelace"	
ch	'A'	65
first	"Ada"	
hashVal	0	
kLargePrime	16908799	
ks	"	

47 * but we thought it might be fun!)

48 */

49 int nameHash(string first, string last){

50 /* This hashing scheme needs two prime numbers, a large prime and a small

51 * prime. These numbers were chosen because their product is less than

52 * $2^{31} - kLargePrime - 1$.

53 */

54 static const int kLargePrime = 16908799;

55 static const int kSmallPrime = 127;

56

57 int hashVal = 0;

58

59 /* Iterate across all the characters in the first name, then the last

60 * name, updating the hash at each step.

61 */

62 for (char c : first){

63 }

64

65

66 }
67
68 }
69
70 return hashVal;

71 }

Once you're confident that you're on the "Step Over" button - and not the "Step Into" or "Step Out" buttons - go and click it! When you do...

Debugger GDB for "Welcome to CS106B"  Threads: #7 Welcome_to_CS10 ... Stopped at breakpoint 1 in thread 7.

Level Function File Line

1	nameHash	Nam...	66
2	qMain	Nam...	31
3	std::Function<int ()>::_M_invoke(std::Any_data const&)	std_f...	302
4	std::function<int ()>::operator()()	std_f...	706
5	QGui::lambda()::operator()(void) const	spl.cpp	20981
6	std::Function<int ()>::operator()()	std_f...	302
7	std::function<int ()>::operator()()	std_f...	706
8	GThreadStd::run	spl.cpp	22491
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514
10	std::_invoke_impl<void, GThreadStd::start()::<lambda()>>(std::_invoke_o...	invok...	60

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results



Sun 12:35 PM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

Welcome to CS106B [master]

- >Welcome to CS106B.pro
- Headers
- Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
- Other files

Name Hash Function Implementation

```
47     * but we thought it might be fun!
48     */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the last
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values of
64         * lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71 }
```

Debugger GDB for "Welcome to CS106B"

Level	Function
1	nameHash
2	qMain
3	std::Function_handle<int ()>
4	std::function<int ()>
5	QGui::lambda()::operator()
6	std::Function_handle<int ()>
7	std::function<int ()>
8	GThreadStd::run
9	GThreadStd::<lambda()>::operator()
10	std::invoke_impl<void, GThreadStd::start()::<lambda()>>(std::function<void ()>, std::optional<GThreadStd::start()::<lambda()>::operator()>)

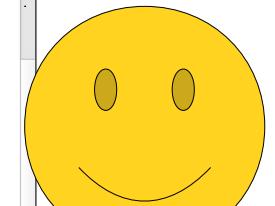
Type to locate (Ctrl+F)

Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

...your window should look something like this.

Value

Name	Value
_for_begin	65 'A'
_for_end	0 '\000'
_for_range	"AdaLovelace"
ch	'a' 97
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"



Sun 12:35 PM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

Welcome to CS106B [master]

- Welcome to CS106B.pro
- Headers
- Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
- Other files

Name Hash Function Implementation

```
47     * but we thought it might be fun!
48     */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the last
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values of
64         * lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71 }
```

Debugger GDB for "Welcome to CS106B"

Level	Function
1	nameHash
2	qMain
3	std::Function_handle<std::function<int ()>>
4	std::function<int ()>
5	QGui::lambda()
6	std::Function_handle<std::function<int ()>>
7	GThreadStd::run
8	GThreadStd::operator()<lambda>
9	std::invoke_impl<void, GThreadStd::start()::<lambda()>>(std::function<void ()>, std::optional<std::function<void ()>)
10	std::invoke

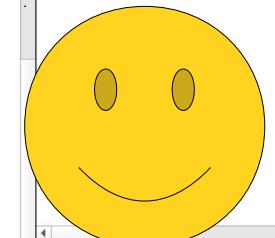
Type to locate (Ctrl...)

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 7 Test Results

Okay! A few things have changed. Let's see what's going on.

Value

Name	Value
_for_begin	65 'A'
_for_end	0 '\000'
_for_range	"AdaLovelace"
ch	'a' 97
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"



Sun 12:35 PM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

Welcome to CS106B [master]

- Welcome to CS106B.pro
- Headers
- Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
- Other files

Line: 67, Col: 9

```
47     * but we thought it might be fun!)
48     */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the last
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values of
64         * lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71 }
```

Debugger GDB for "Welcome to CS106B"
Level Function
1 nameHash
2 qMain
3 std::Function_handle<std::function<int ()>>
4 std::function<int ()>
5 QGui::lambda()
6 std::Function_handle<std::function<int ()>>
7 std::function<int ()>
8 GThreadStd::run
9 GThreadStd::operator<<(std::basic_ostream<char> &, GThreadStd::start()::lambda())
10 std::invoke_impl<void, GThreadStd::start()::lambda()>(std::function<void ()>, GThreadStd::start()::lambda())

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

First, notice that our helpful Yellow Arrow friend is now pointing at line 67.

Yellow Arrow friend: A yellow smiley face icon with a black outline, positioned next to the line number 67 in the code editor.

Variables Table: A table showing the current state of variables in the debugger. The columns are Name, Value, and Type.

Name	Value	Type
_for_begin	65 'A'	
_for_end	0 '\000'	
_for_range	"AdaLovelace"	
ch	'a'	97
first	"Ada"	
hashVal	0	
kLargePrime	16908799	
kSmallPrime	127	
last	"Lovelace"	

Sun 12:35 PM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

Welcome to CS106B [master]

- Welcome to CS106B.pro
- Headers
- Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
- Other files

Line: 67, Col: 9

```
47     * but we thought it might be fun!)
48     */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the last
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values of
64         * lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71 }
```

Debugger GDB for "Welcome to CS106B"

Level	Function
1	nameHash
2	qMain
3	std::Function_handle<std::function<int ()>>
4	std::function<int ()>
5	QGui::lambda()
6	std::Function_handle<std::function<int ()>>
7	GThreadStd::run
8	GThreadStd::operator()<lambda>
9	std::invoke_impl<void, GThreadStd::start()<lambda()>>(std::function<void ()>, GThreadStd::start()<lambda()>)
10	std::invoke_impl<void, GThreadStd::start()<lambda()>>(std::function<void ()>, GThreadStd::start()<lambda()>)

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

We're now at the line right after the one where we stopped. You just ran a single line of the program! Pretty cool!

Smiley face icon in the status bar.

Variables table:

Name	Value
_for_begin	65 'A'
_for_end	0 '\000'
_for_range	"AdaLovelace"
ch	'a' 97
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

Sun 12:35 PM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

Welcome to CS106B [master]

- Welcome to CS106B.pro
- Headers
- Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
- Other files

Name Hash Function Implementation

```
47     * but we thought it might be fun!
48     */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the last
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values of
64         * lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71 }
```

Debugger GDB for "Welcome to CS106B"

Level	Function
1	nameHash
2	qMain
3	std::function<int ()>
4	std::function<void (const QString&)>
5	QGui::lambda()::operator()<void (const QString&)>
6	std::function<int ()>
7	std::function<int ()>
8	GThreadStd::run
9	GThreadStd::operator()<void (const QString&)>
10	std::invoke_impl<void, GThreadStd::start()::<lambda()>>(std::function<void (const QString&)>, GThreadStd::start()::<lambda()>)

Type to locate (Ctrl...)

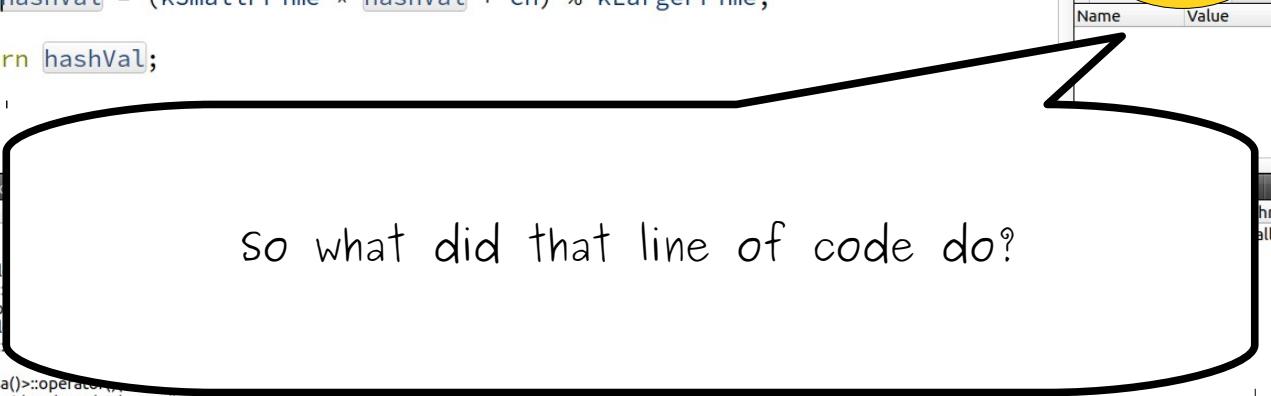
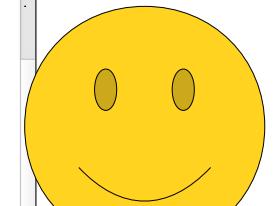
Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

Line: 67, Col: 9

Name Value

Name	Value
_for_begin	65 'A'
_for_end	0 '\000'
_for_range	"AdaLovelace"
ch	'a' 97
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

So what did that line of code do?



Sun 12:35 PM
NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

Welcome to CS106B [master]

- Welcome to CS106B.pro
- Headers
- Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
- Other files

Name Hash

Line: 67, Col: 9

Name	Value
_for_begin	65 'A'
_for_end	0 '\000'
_for_range	"AdaLovelace"
ch	'a'
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

```
47     * but we thought it might be fun!
48     */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the last
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values of
64         * lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashval = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71 }
```

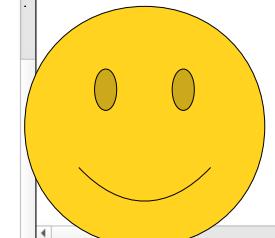
Debugger GDB for "Welcome to CS106B"

Level Function

- 1 nameHash
- 2 qMain
- 3 std::Function_handle<std::function<int ()>>
- 4 std::function<int ()>
- 5 QGui::lambda()::operator()
- 6 std::Function_handle<std::function<int ()>>
- 7 std::function<int ()>
- 8 GThreadStd::run
- 9 GThreadStd::operator()
- 10 std::invoke_impl<void, GThreadStd::start()::lambda()>::operator()

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

This line converts ch to lower case. The tolower function takes in a character and returns a lower-case version of it, so this overwrites ch with a lower-case version of itself.



Activities Qt Creator ▾

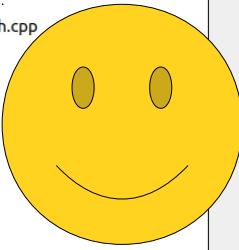
Sun 12:35 PM
NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

- Welcome to CS106B [master]
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - SRC
 - NameHash.cpp
 - Other files

You can actually see this by looking at the values panel over on the side!



55
56
57 int hashVal = 0;
58
59 /* Iterate across all the characters in the first name, then the last
60 * name, updating the hash at each step.
61 */
62 for (char ch: first + last) {
63 /* Convert the input character to lower case. The numeric values of
64 * lower-case letters are always less than 127.
65 */
66 ch = tolower(ch);
67 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68 }
69 return hashVal;
70
71 }

Debugger GDB for "Welcome to CS106B" Threads: #7 Welcome_to_CS10 Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	67	1	...ring)	...eHash.cpp	66	...555892674	(all)		
2	qMain	Nam...	31								
3	std::function<int ()>::_M_invoke(std::any_data const&)	std_f...	302								
4	std::function<int ()>::operator()()	std_f...	706								
5	QGui::lambda()::operator()(void) const	spl.cpp	20981								
6	std::function<int ()>::operator(), QtGui::startBackgroundEventLoop(GThunkin...	std_f...	302								
7	std::function<int ()>::operator()()	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()::<lambda()>>(std::_invoke_o...	invok...	60								

Type to locate (Ctrl...) 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 7 Test Results

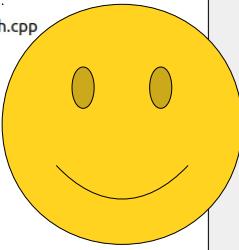
Activities Qt Creator ▾

Sun 12:35 PM
NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

- Welcome to CS106B [master]
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - silicon-large.png
 - res
 - SRC
 - NameHash.cpp
 - Other files



Notice that the value associated with ch has changed from 'A' to 'a' - it's now in lower-case!

```
55
56
57 int hashVal = 0;
58
59 /* Iterate across all the characters in the first name, then the last
   * name, updating the hash at each step.
   */
60 for (char ch: first + last) {
61     /* Convert the input character to lower case. The numeric values of
       * lower-case letters are always less than 127.
       */
62     ch = tolower(ch);
63     hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
64 }
65 return hashVal;
66
67 }
```

Debugger GDB for "Welcome to CS106B" Threads: #7 Welcome_to_CS10 Stopped: "end-stepping-range".

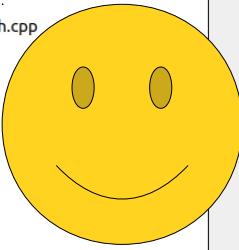
Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	67	1	...ring)	...eHash.cpp	66	...555892674	(all)		
2	qMain	Nam...	31								
3	std::function<int ()>::_M_invoke(std::any_data const&)	std_f...	302								
4	std::function<int ()>::operator()()	std_f...	706								
5	QGui::lambda()::operator()(void) const	spl.cpp	20981								
6	std::function<int ()>::operator()()	std_f...	302								
7	std::function<int ()>::operator()()	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()::<lambda()>>(std::_invoke_o...	invok...	60								

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

Sun 12:35 PM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

If you'll notice, this value is in red while all the other values are in black.



```
File Edit Build Debug Analyze Tools Window Help
```

Projects

- Welcome to CS106B [master]
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - SRC
 - NameHash.cpp
 - Other files

55
56
57 int hashVal = 0;
58
59 /* Iterate across all the characters in the first name, then the last
60 * name, updating the hash at each step.
61 */
62 for (char ch: first + last) {
63 /* Convert the input character to lower case. The numeric values of
64 * lower-case letters are always less than 127.
65 */
66 ch = tolower(ch);
67 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68 }
69 return hashVal;
70
71 }

Debugger GDB for "Welcome to CS106B" Threads: #7 Welcome_to_CS10 Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	67	1	...ring)	...eHash.cpp	66	...555892674	(all)		
2	qMain	Nam...	31								
3	std::function<int ()>::_M_invoke(std::any_data const&)	std_f...	302								
4	std::function<int ()>::operator()()	std_f...	706								
5	QtGui::lambda()::operator()(void) const	spl.cpp	20981								
6	std::function<int ()>::operator()()	std_f...	302								
7	std::function<int ()>::operator()()	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()::<lambda()>>(std::_invoke_o...	invok...	60								

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

Col: 9

Name	Value
_for_begin	65 'A'
_for_end	0 '\000'
_for_range	AdaLovelace
ch	'a'
First	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

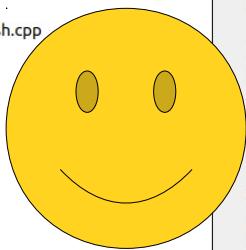
Sun 12:35 PM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

- Welcome to CS106B [master]
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - silicon-large.png
 - res
 - SRC
 - NameHash.cpp
 - Other files



This indicates that the value here has changed since the previous step. This is a really useful way to keep track of what's changing as you run the program.

```
55
56
57 int hashVal = 0;
58
59 /* Iterate across all the characters in the first name, then the last
   * name, updating the hash at each step.
   */
60
61 for (char ch: first + last) {
62     /* Convert the input character to lower case. The numeric values of
       * lower-case letters are always less than 127.
       */
63     ch = tolower(ch);
64     hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
65 }
66
67 return hashVal;
68
69
70
71 }
```

Name	Value
_for_begin	65 'A'
_for_end	0 '\000'
_for_range	AdaLovelace
ch	'a' 97
First	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

Debugger GDB for "Welcome to CS106B"

Threads: #7 Welcome_to_CS10 Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	67	1	...ring)	...eHash.cpp	66	...555892674	(all)		
2	qMain	Nam...	31								
3	std::function<int (), int (*)()>::_M_invoke(std::any_data const&)	std_f...	302								
4	std::function<int ()>::operator()()	std_f...	706								
5	QtGui::lambda()>::operator()(void) const	spl.cpp	20981								
6	std::function<int (), QtGui::startBackgroundEventLoop(GThunkin...	std_f...	302								
7	std::function<int ()>::operator()()	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()::<lambda()>>(std::_invoke_o...	invok...	60								

Activities Qt Creator ▾

Sun 12:35 PM
NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

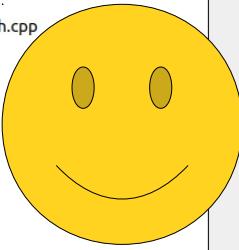
- Welcome to CS106B [master]
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - SRC
 - NameHash.cpp
 - Other files

Col: 9

Name Value

__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'a'
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

Now, let's take a look at line 67, where we are right now.



55
56
57 int hashVal = 0;
58
/* Iterate across all the characters in the first name, then the last
* name, updating the hash at each step.
*/
59 for (char ch: first + last) {
60 /* Convert the input character to lower case. The numeric values of
61 * lower-case letters are always less than 127.
*/
62 ch = tolower(ch);
63 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
64 }
65
66
67 }
68 return hashVal;
69
70 }

Debugger GDB for "Welcome to CS106B" Threads: #7 Welcome_to_CS10 Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	67	1	...ring)	...eHash.cpp	66	...555892674	(all)		
2	qMain	Nam...	31								
3	std::function<int ()>::_M_invoke(std::any_data const&)	std_f...	302								
4	std::function<int ()>::operator()()	std_f...	706								
5	QtGui::lambda()::operator()(void) const	spl.cpp	20981								
6	std::function<int ()>::operator()()	std_f...	302								
7	std::function<int ()>::operator()()	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()::<lambda()>>(std::_invoke_o...	invok...	60								

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

Activities Qt Creator ▾

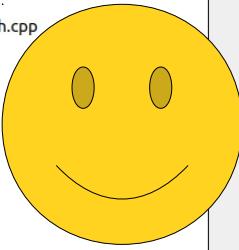
Sun 12:35 PM
NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

- Welcome to CS106B [master]
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - silicon-large.png
 - res
 - SRC
 - NameHash.cpp
 - Other files

Col: 9



Not gonna lie, this is a pretty dense line of code. It performs some weird sort of mathematical calculation on a bunch of different values.

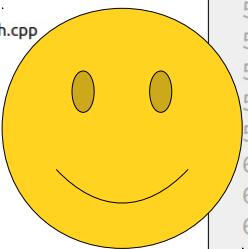
```
55
56
57 int hashVal = 0;
58
59 /* Iterate across all the characters in the first name, then the last
   * name, updating the hash at each step.
   */
60 for (char ch: first + last) {
61     /* Convert the input character to lower case. The numeric values of
       * lower-case letters are always less than 127.
       */
62     ch = tolower(ch);
63     hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
64 }
65 return hashVal;
66 }
```

Debugger GDB for "Welcome to CS106B" Threads: #7 Welcome_to_CS10 Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	67	1	...ring)	...eHash.cpp	66	...555892674	(all)		
2	qMain	Nam...	31								
3	std::function<int ()>::_M_invoke(std::any_data const&)	std_f...	302								
4	std::function<int ()>::operator()()	std_f...	706								
5	QtGui::lambda()::operator()(void) const	spl.cpp	20981								
6	std::function<int ()>::operator(), QtGui::startBackgroundEventLoop(GThunkin...	std_f...	302								
7	std::function<int ()>::operator()()	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()::<lambda()>>(std::_invoke_o...	invok...	60								

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

Fundamentally, though, it's just computing some weird function of some values and stashing it into `hashVal`.



```
int hashVal = 0;

/* Iterate across all the characters in the first name, then the last
 * name, updating the hash at each step.
 */
for (char ch: first + last) {
    /* Convert the input character to lower case. The numeric values of
     * lower-case letters are always less than 127.
     */
    ch = tolower(ch);
    hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
}
return hashVal;
```

Debugger GDB for "Welcome to CS106B" Threads: #7 Welcome_to_CS10 Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	67	1	...ring)	...eHash.cpp	66	...555892674	(all)		
2	qMain	Nam...	31								
3	std::function<int ()>::_M_invoke(std::any_data const&)	std_f...	302								
4	std::function<int ()>::operator()()	std_f...	706								
5	QtGui::lambda()::operator()(void) const	spl.cpp	20981								
6	std::function<int ()>::operator()()	std_f...	302								
7	std::function<int ()>::operator()()	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()::<lambda()>>(std::_invoke_o...	invok...	60								

Type to locate (Ctrl...) 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 7 Test Results 8

Activities Qt Creator ▾

Sun 12:35 PM
NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

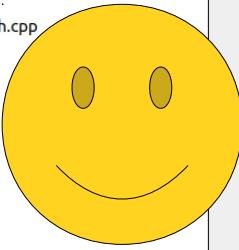
- Welcome to CS106B [master]
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - SRC
 - NameHash.cpp
 - Other files

Col: 9

Name Value

__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'a'
first	"Ada"
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

Let's go run that line of code and see what happens!

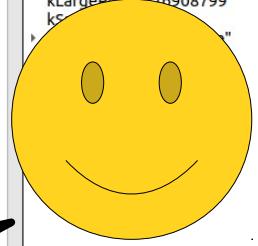


55
56
57 int hashVal = 0;
58
59 /* Iterate across all the characters in the first name, then the last
60 * name, updating the hash at each step.
61 */
62 for (char ch: first + last) {
63 /* Convert the input character to lower case. The numeric values of
64 * lower-case letters are always less than 127.
65 */
66 ch = tolower(ch);
67 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68 }
69 return hashVal;
70 }
71 }

Debugger GDB for "Welcome to CS106B" Threads: #7 Welcome_to_CS10 Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	67	1	...ring)	...eHash.cpp	66	...555892674	(all)		
2	qMain	Nam...	31								
3	std::function<int ()>::_M_invoke(std::any_data const&)	std_f...	302								
4	std::function<int ()>::operator()()	std_f...	706								
5	QtGui::lambda()::operator()(void) const	spl.cpp	20981								
6	std::function<int ()>::operator()()	std_f...	302								
7	std::function<int ()>::operator()()	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()::<lambda()>>(std::_invoke_o...	invok...	60								

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results



Hover over the "Step Over" button, confirm that the button you're clicking really is "Step Over," and click it! When you do...



Sun 12:39 PM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

Welcome to CS106B [master]

- Welcome to CS106B.pro
- Headers
- Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
- Other files

NameHash.cpp

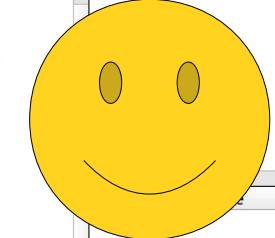
```
47     * but we thought it might be fun!
48     */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the last
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values of
64         * lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71 }
```

Debugger GDB

Level	Function	File	Line
1	nameHash	spl.cpp	706
2	qMain	spl.cpp	22491
3	std::function<void()>::operator()() const	spl.cpp	22514
4	std::function<void()>::operator()() const	spl.cpp	60
5	QThreadStd::run		
6	GThreadStd::<lambda()>::operator()() const		
7	std::invoke_impl<void, GThreadStd::start()::<lambda()>>(std::function<void()>)		
8	GThreadStd::start()		
9	GThreadStd::<lambda()>::operator()() const		
10	std::function<void()>::operator()() const		

Type to locate (Ctrl...)

Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results


... you'll end up with something like this!

Name Value

Name	Value
_for_begin	65 'A'
_for_end	0 '\000'
_for_range	"AdaLovelace"
ch	'a' 97
first	"Ada"
hashVal	97
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

Sun 12:39 PM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

Welcome to CS106B [master]

- Welcome to CS106B.pro
- Headers
- Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
- Other files

NameHash.cpp

```
47     * but we thought it might be fun!
48     */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the last
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values of
64         * lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71 }
```

Debugger GDB for Welc...106B

Level	Function	File	Line
1	nameHash	spl.cpp	706
2	qMain	spl.cpp	22491
3	std::function<void()>::operator()() const	spl.cpp	22514
4	std::function<void()>::operator()() const	spl.cpp	60
5	QThreadStd::run		
6	GThreadStd::<lambda()>::operator()() const		
7	std::invoke_impl<void, GThreadStd::start()::<lambda()>>(std::function<void()>)		
8	GThreadStd::start()		
9	GThreadStd::<lambda()>::operator()() const		
10	std::function<void()>::operator()() const		

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

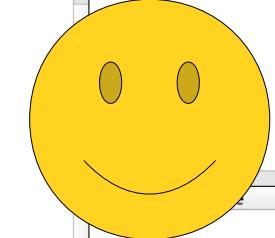
Line: 62, Col: 5

Name Value

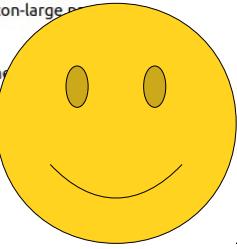
Name	Value
_for_begin	65 'A'
_for_end	0 '\000'
_for_range	"AdaLovelace"
ch	'a' 97
first	"Ada"
hashVal	97
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

Views Threads (all)

Let's see what's changed.



First, notice that the value stored in `hashVal` changed to 97. We know that it changed because the value is in red, and we know that nothing else changed because nothing else is in red!



```
File Edit Build Debug Analyze Tools Window Help
Projects
  Welcome to CS106B [master]
    Welcome to CS106B.pro
    Sources
      lib
        StanfordCPPLib
          spl.cpp
          addr2line.exe
          addr2line64.exe
          iconstrip.png
          silicon-large.p
      res
      src
        Name.h
        Other files
        
```

```
52
53
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the last
60      * name, updating the hash at each step.
61      */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values of
64          * lower-case letters are always less than 127.
65          */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71 
```

Debugger GDB for "Welcome to CS106B" Threads: #7 Welcome_to_CS10 Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	62	1	...ring)	...eHash.cpp	66	...555892674	(all)		
2	qMain	Nam...	31								
3	std::function<int (), int (*)()>::_M_invoke(std::any_data const&)	std_f...	302								
4	std::function<int ()>::operator()()	std_f...	706								
5	QtGui::lambda()>::operator()(void) const	spl.cpp	20981								
6	std::function<int (), QtGui::startBackgroundEventLoop(GThunkin...	std_f...	302								
7	std::function<int ()>::operator()()	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()::<lambda()>>(std::_invoke_o...	invok...	60								

Type to locate (Ctrl...) 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 8 Test Results

Name	Value
_for_begin	65 'A'
_for_end	0 '\000'
_for_range	"AdaLovelace"
ch	'a'
hashVal	97
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

Sun 12:39 PM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

Line: 62, Col: 5

Name	Value
_for_begin	65 'A'
_for_end	0 '\000'
_for_range	"AdaLovelace"
ch	'a' 97
first	"Ada" 97
hashVal	16908799
kLargePrime	127
kSmallPrime	127
last	"Lovelace"

47 * but we thought it might be fun!
 48 */
 49 int nameHash(string first, string last){
 50 /* This hashing scheme needs two prime numbers, a large prime and a small
 51 * prime. These numbers were chosen because their product is less than
 52 * 2^31 - 1.
 53 */
 54 stat
 55 stat
 56 int
 57 /* I
 58 * n
 59 */
 60 for (char
 61 /* Convert the input character to lower case. The number
 62 * lower-case letters are always less than 127.
 63 */
 64 ch = tolower(ch);
 65 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
 66 }
 67 return hashVal;
 68 }
 69
 70
 71 }

Second, notice that we're back up at the top of the for loop, since that's where the yellow arrow is pointing. We ended up back here because this is the next line that gets executed.

Type

Debugger GDB for "Welcome to CS106B"

Threads: #7 Welcome_to_CS10 Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	62	1	...ring)	...eHash.cpp	66	...555892674	(all)		
2	qMain	Nam...	31								
3	std::function<int ()>::_M_invoke(std::any_data const&)	std_f...	302								
4	std::function<int ()>::operator()()	std_f...	706								
5	QGui::lambda()::operator()(void) const	spl.cpp	20981								
6	std::function<int ()>::operator()()	std_f...	302								
7	std::function<int ()>::operator()()	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()::<lambda()>::operator()>(std::_invoke_o...	invok...	60								

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

Sun 12:39 PM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

Welcome to CS106B [master]

- Welcome to CS106B.pro
- Headers
- Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
- Other files

Line: 62, Col: 5

Name Value

__for_begin	65 'A'
__for_end	0 '\000'
__for_range	"AdaLovelace"
ch	'a' 97
first	"Ada"
hashVal	97
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

47 * but we thought it might be fun!)

48 */

49 int nameHash(string first, string last){

50 /* This hashing scheme needs two prime numbers, a large prime and a small

51 * prime. These numbers were chosen because their product is less than

52 * $2^{31} - kLargePrime - 1$.

53 */

54 static const int kLargePrime = 16908799;

55 static const int kSmallPrime = 127;

56

57

58

59

60

61

62

63

64

65

66 for (int i = 0; i < first.length(); i++) {

67 /* Convert the input character to lower case. The number of

68 * lower-case letters are always less than 127.

69 */

70 ch = tolower(ch);

71 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;

72 }

73 return hashVal;

74 }

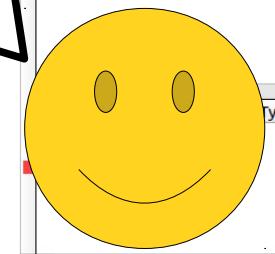
We just single-stepped through a single iteration
of that loop! Pretty cool!

Debugger GDB for "Welcome to CS106B"

Threads: #7 Welcome_to_CS10 Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	62	1	...ring)	...eHash.cpp	66	...555892674	(all)		
2	qMain	Nam...	31								
3	std::function<int ()>::_M_invoke(std::any_data const&)	std_f...	302								
4	std::function<int ()>::operator()()	std_f...	706								
5	QGui::lambda()::operator()(void) const	spl.cpp	20981								
6	std::function<int ()>::operator(), QtGui::startBackgroundEventLoop(GThunkin...	std_f...	302								
7	std::function<int ()>::operator()()	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
10	std::invoke_implementation<void, GThreadStd::start()::<lambda()>::operator_o...	invok...	60								

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results



Sun 12:39 PM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - silicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

Line: 62, Col: 5

Name	Value
_for_begin	65 'A'
_for_end	0 '\000'
_for_range	"AdaLovelace"
ch	'a' 97
first	"Ada" 97
hashVal	16908799
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

47 * but we thought it might be fun!)

48 */

49 int nameHash(string first, string last){

50 /* This hashing scheme needs two prime numbers, a large prime and a small prime. These numbers were chosen because their product is less than $2^{31} - kLargePrime - 1$.

51 */

52 static const int kLargePrime = 16908799;

53 static const int kSmallPrime = 127;

54

55

56

57

58

59

60

61

62

63

64

65

66 ch = tolower(ch);

67 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;

68 }

69 return hashVal;

70 }

71 }

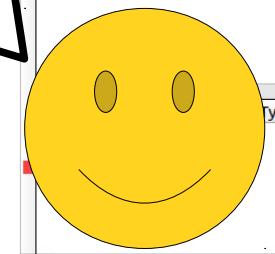
Let's go do it again!

Debugger GDB for "Welcome to CS106B"

Threads: #7 Welcome_to_CS10 Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	62	1	...ring)	...eHash.cpp	66	...555892674	(all)		
2	qMain	Nam...	31								
3	std::function<int ()>::_M_invoke(std::any_data const&)	std_f...	302								
4	std::function<int ()>::operator()()	std_f...	706								
5	QGui::lambda()::operator()(void) const	spl.cpp	20981								
6	std::function<int ()>::operator()()	std_f...	302								
7	std::function<int ()>::operator()()	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()()	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()::<lambda()>::operator()>(std::_invoke_o...	invok...	60								

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results



Sun 12:39 PM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

Welcome to CS106B [master]

- Welcome to CS106B.pro
- Headers
- Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - silicon-large.png
 - res
 - src
 - NameHash.cpp
- Other files

Line: 62, Col: 5

Name	Value
_for_begin	65 'A'
_for_end	0 '\000'
_for_range	"AdaLovelace"
ch	'a' 97
first	"Ada"
hashVal	97
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

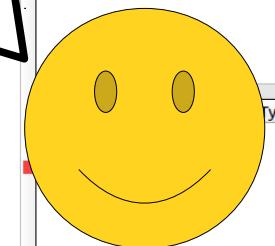
```
47     * but we thought it might be fun!)
48     */
49     int nameHash(string first, string last){
50         /* This hashing scheme needs two prime numbers, a large prime and a small
51         * prime. These numbers were chosen because their product is less than
52         * 2^31 - kLargePrime - 1.
53         */
54         static const int kLargePrime = 16908799;
55         static const int kSmallPrime = 127;
56
57         int hashVal = 1;
58
59         /* Initialize hashVal to 1. This is the identity element for modular
60         * addition. It also handles the case where both strings are empty.
61         */
62         for (int i = 0; i < first.length(); i++) {
63             /* Convert the input character to lower case. The numbers for
64             * lower-case letters are always less than 127.
65             */
66             ch = tolower(ch);
67             hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68         }
69         return hashVal;
70     }
71 }
```

Again, move your mouse over the Step Over button (and make sure it says "Step Over" and not something else!), then click it.

Debugger GDB for "Welcome to CS106B"  Threads: #7 Welcome_to_CS10 Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	62	1	...ring)	...eHash.cpp	66	...555892674	(all)		
2	qMain	Nam...	31								
3	std::function<int ()>::_M_invoke(std::any_data const&)	std_f...	302								
4	std::function<int ()>::operator()()	std_f...	706								
5	QGui::lambda()::operator()(void) const	spl.cpp	20981								
6	std::function<int ()>::operator()()	std_f...	302								
7	std::function<int ()>::operator()()	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
10	std::invoke_impl<void, GThreadStd::start()::<lambda()>::operator()>(std::invok...	invok...	60								

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results



Sun 12:42 PM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

- Welcome to CS106B [master]
 - >Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

```

47     * but we thought it might be fun!
48     */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 97;
58
59     /* Initialize hashVal to 97. This is a good starting point for a hash value.
60     * Note that all lower-case letters have ASCII values less than 127.
61     */
62     for (char ch : first)
63         /* Convert the input character to lower case. The numbers for lower-case
64         * letters are always less than 127.
65         */
66     {
67         ch = tolower(ch);
68         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
69     }
70     return hashVal;
71 }
```

Debugger GDB for "Welcome to CS106B"

Threads: #7 Welcome_to_CS10 Stopped at breakpoint 1 in thread 7.

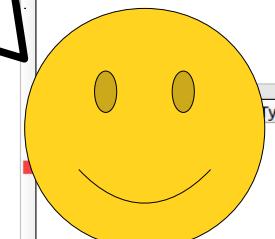
Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring)	...eHash.cpp	66	...555892674	(all)		
2	qMain	Nam...	31								
3	std::function<int ()>::_M_invoke(std::any_data const&)	std_f...	302								
4	std::function<int ()>::operator()()	std_f...	706								
5	QGui::lambda()::operator()(void) const	spl.cpp	20981								
6	std::function<int ()>::operator()()	std_f...	302								
7	std::function<int ()>::operator()()	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
10	std::invoke_impl<void, GThreadStd::start()::<lambda()>::operator()>(std::invok...	invok...	60								

Type to locate (Ctrl...) 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 7 Test Results

Variables

Name	Value
_for_begin	100'd'
_for_end	0'\000'
_for_range	Ada Lovelace
ch	'd'
first	"Ada"
hashVal	97
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

Note: Now we're here! Notice that ch now has the value 'd', which is the second letter of the name Ada.



Sun 12:42 PM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

Welcome to CS106B [master]

- Welcome to CS106B.pro
- Headers
- Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
- Other files

Line: 66, Col: 9

Name Value

__for_begin	100'd'
__for_end	0'\000'
__for_range	"AdaLovelace"
ch	'd'
first	"Ada"
hashVal	97
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

47 * but we thought it might be fun!)

48 */

49 int nameHash(string first, string last){

50 /* This hashing scheme needs two prime numbers, a large prime and a small

51 * prime. These numbers were chosen because their product is less than

52 * $2^{31} - kLargePrime - 1$.

53 */

54 static const int kLargePrime = 16908799;

55 static const int kSmallPrime = 127;

56

57 int hash(

58 /* I

59 * n

60 */

61 for(

62 /* Convert the input character to lower case. The number

63 * lower-case letters are always less than 127.

64 */

65

66 ch = tolower(ch);

67 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;

68 }

69 return hashVal;

70 }

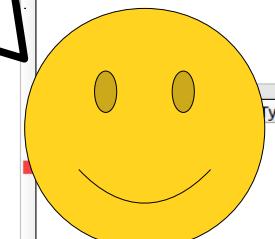
71 }

Go click "Step Over" again to run this line of code.

Debugger GDB for "Welcome to CS106B"  Threads: #7 Welcome_to_CS10 Stopped at breakpoint 1 in thread 7.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	66	1	...ring)	...eHash.cpp	66	...555892674	(all)		
2	qMain	Nam...	31								
3	std::function<int ()>::_M_invoke(std::Any_data const&)	std_f...	302								
4	std::function<int ()>::operator()()	std_f...	706								
5	QtGui::lambda()>::operator()(void) const	spl.cpp	20981								
6	std::function<int ()>::operator()()	std_f...	302								
7	std::function<int ()>::operator()()	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()()	spl.cpp	22514								
10	std::invoke_impl<void, GThreadStd::start()::<lambda()>>(std::invok...	invok...	60								

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results



Sun 12:43 PM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

Welcome to CS106B [master]

- Welcome to CS106B.pro
- Headers
- Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - silicon-large.png
 - res
 - src
 - NameHash.cpp
- Other files

Line: 67, Col: 9

Name	Value
_for_begin	100'd'
_for_end	0'000'
_for_range	"AdaLovelace"
ch	'd'
first	"Ada"
hashVal	97
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

47 * but we thought it might be fun!)

48 */

49 int nameHash(string first, string last){

50 /* This hashing scheme needs two prime numbers, a large prime and a small

51 * prime. These numbers were chosen because their product is less than

52 * $2^{31} - kLargePrime - 1$.

53 */

54 static const int kLargePrime = 16908799;

55 static const int kSmallPrime = 127;

56

57

58

59 int

60 /* I

61 * n

62 for (char ch : last)

63 /* Convert the input character to lower case. The number

64 * lower-case letters are always less than 127.

65 */

66 ch = tolower(ch);

67 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;

68 }

69 return hashVal;

70 }

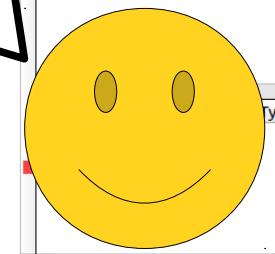
71 }

You should be here now. Notice that none of the values changed. That makes sense, since all we did was convert a lower-case 'd' to a lower-case 'd'.

Debugger GDB for "Welcome to CS106B" Threads: #7 Welcome_to_CS10 Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	67	1	...ring)	...eHash.cpp	66	...555892674	(all)		
2	qMain	Nam...	31								
3	std::function<int ()>::_M_invoke(std::any_data const&)	std_f...	302								
4	std::function<int ()>::operator()()	std_f...	706								
5	QGui::lambda()::operator()(void)	spl.cpp	20981								
6	std::function<int ()>::operator()()	std_f...	302								
7	std::function<int ()>::operator()()	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void)	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()::<lambda()>::operator()>(std::_invoke_o...	invok...	60								

Type to locate (Ctrl...) 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 7 Test Results



Sun 12:43 PM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

Welcome to CS106B [master]

- Welcome to CS106B.pro
- Headers
- Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
- Other files

Line: 67, Col: 9

Name Value

_for_begin	100'd'
_for_end	0'\000'
_for_range	"AdaLovelace"
ch	'd' 100
first	"Ada"
hashVal	97
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

47 * but we thought it might be fun!)

48 */

49 int nameHash(string first, string last){

50 /* This hashing scheme needs two prime numbers, a large prime and a small

51 * prime. These numbers were chosen because their product is less than

52 * $2^{31} - kLargePrime - 1$.

53 */

54 static const int kLargePrime = 16908799;

55 static const int kSmallPrime = 127;

56

57 int

58 /* I

59 * n

60 */

61 for (char ch : first)

/* Convert the input character to lower case. The number of

lower-case letters are always less than 127.

*/

62 ch = tolower(ch);

63 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;

64 }

65 return hashVal;

66 }

67 }

68 }

69 }

70 }

71 }

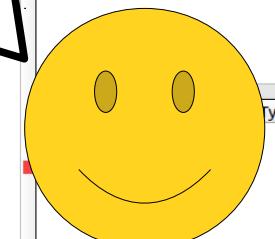
Now, click "Step Over" one more time.

Debugger GDB for "Welcome to CS106B"  Threads: #7 Welcome_to_CS10 Stopped: "end-stepping-range".

Level Function File Line Number Function File Line Address Condition Ignore Threads

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	67	1	...ring)	...eHash.cpp	66	...555892674	(all)		
2	qMain	Nam...	31								
3	std::function<int ()>::_M_invoke(std::any_data const&)	std_f...	302								
4	std::function<int ()>::operator()()	std_f...	706								
5	QGui::lambda()::operator()(void)	spl.cpp	20981								
6	std::function<int ()>::operator()()	std_f...	302								
7	std::function<int ()>::operator()()	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()()	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()::<lambda()>::operator()>(std::_invoke_o...	invok...	60								

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results



Sun 12:43 PM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

Welcome to CS106B [master]

- >Welcome to CS106B.pro
- Headers
- Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - silicon-large.png
 - res
 - SRC
 - NameHash.cpp
- Other files

Line: 62, Col: 5

Name	Value
_for_begin	100'd'
_for_end	0'000'
_for_range	"AdaLovelace"
ch	'd' 100
first	"Ada"
hashVal	???
kLargePrime	16908799
kSmallPrime	127
last	"Lovelace"

47 * but we thought it might be fun!)

48 */

49 int nameHash(string first, string last){

50 /* This hashing scheme needs two prime numbers, a large prime and a small

51 * prime. These numbers were chosen because their product is less than

52 * $2^{31} - kLargePrime - 1$.

53 */

54 static const int kLargePrime = 16908799;

55 }

56 return hashVal;

57 }

58 }

59 }

60 }

61 }

62 }

63 }

64 }

65 }

66 }

67 }

68 }

69 }

70 }

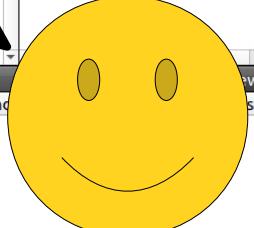
71 }

You'll now be at this point in the program. We've covered up the value of hashVal in this image, because at this point you should be able to see what hashVal is by reading the value in the side pane. **This is the special value we want you to tell us when submitting the assignment!**

Debugger GDB for "Welcome to CS106B" Threads: #7 Welcome_to_CS10 Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition
1	nameHash	Nam...	62	1	...ring)	...eHash.cpp	66	...555892674	
2	qMain	Nam...	31						
3	std::function<int ()>::_M_invoke(std::any_data const&)	std_f...	302						
4	std::function<int ()>::operator()()	std_f...	706						
5	QtGui::lambda()::operator()(void) const	spl.cpp	20981						
6	std::function<int ()>::operator()()	std_f...	302						
7	std::function<int ()>::operator()()	std_f...	706						
8	GThreadStd::run	spl.cpp	22491						
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514						
10	std::invoke_impl<void, GThreadStd::start()::<lambda()>::operator()>(std::invok...	invok...	60						

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results



Sun 12:43 PM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - silicon-large.png
 - res
 - SRC
 - NameHash.cpp
- Other files

47 * but we thought it might be fun!)
48 */
49 int nameHash(string first, string last){
50 /* This hashing scheme needs two prime numbers, a large prime and a small
 * prime. These numbers were chosen because their product is less than
 * $2^{31} - kLargePrime - 1$.
51
52
53
54
55
56
57
58
59
59 */
60
61
62
63 /* Convert the input character to lower case.
 * lower-case letters are always less than 127.
 */
64
65
66 ch = tolower(ch);
67 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68 }
69 return hashVal;
70
71 }

To finish up this section on the debugger, we'd like to show you two last little techniques that you might find useful when debugging programs.

Debugger GDB for "Welcome to CS106B"

Threads: #7 Welcome_to_CS10 Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	62	1	...ring)	...eHash.cpp	66	...555892674	(all)		
2	qMain	Nam...	31								
3	std::function<int ()>::_M_invoke(std::any_data const&)	std_f...	302								
4	std::function<int ()>::operator()() const	std_f...	706								
5	QGui::lambda()::operator()(void) const	spl.cpp	20981								
6	std::function<int ()>::operator(), QGui::startBackgroundEventLoop(GThunkin...	std_f...	302								
7	std::function<int ()>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()::<lambda()>>(std::_invoke_o...	invok...	60								

Type to locate (Ctrl...) 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 7 Test Results

Activities Qt Creator

Sun 12:43 PM
NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects Welcome to CS106B [master]
Welcome to CS106B.pro
Headers
Sources
lib
StanfordCPPLib
spl.cpp
addr2line.exe
addr2line64.exe
iconstrip.png
silicon-large.png
res
SRC
NameHash.cpp
Other files

NameHash.cpp

```
47     * but we thought it might be fun!
48     */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53 }
```

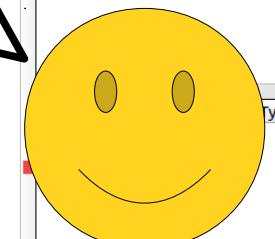
To start this off, click on the the breakpoint that we set earlier in the program. If you do...

```
63     /* Convert the input character to lower case.
64     * lower-case letters are always less than 127.
65     */
66     ch = tolower(ch);
67     hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68 }
69 return hashVal;
70 }
71 }
```

Debugger GDB for "Welcome to CS106B" Threads: #7 Welcome_to_CS10 Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	62	1	...ring)	...eHash.cpp	66	...555892674	(all)		
2	qMain	Nam...	31								
3	std::function<int ()>::_M_invoke(std::_Any_data const&)	std_f...	302								
4	std::function<int ()>::operator()()	std_f...	706								
5	QGui::lambda()::operator()(void) const	spl.cpp	20981								
6	std::function<int ()>::operator(), QtGui::startBackgroundEventLoop(GThunkin...	std_f...	302								
7	std::function<int ()>::operator()()	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()::<lambda()>::operator()>(std::_invoke_o...	invok...	60								

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results



Sun 12:46 PM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

Welcome to CS106B [master]

- >Welcome to CS106B.pro
- Headers
- Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - silicon-large.png
 - res
 - SRC
 - NameHash.cpp
- Other files

Line: 62, Col: 5

Name	Value
_for_begin	100'd'
_for_end	0'\000'
_for_range	"AdaLovelace"
ch	'd'
first	"Ada"
hashVal	??? ???
kLargePrime	109951162799
kSmallPrime	127
last	"Lovelace"

... it should clear the breakpoint. Now, if we were to run this program again in debug mode, it would not stop at this point, since nothing's telling it to!

/* Convert the input character to lower case.
 * lower-case letters are always less than 127.
ch = tolower(ch);
hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
}
return hashVal;

Debugger GDB for "Welcome to CS106B" Threads: #7 Welcome_to_CS10 Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	62								
2	qMain	Nam...	31								
3	std::function<int ()>::_M_invoke(std::any_data const&)	std_f...	302								
4	std::function<int ()>::operator()() const	std_f...	706								
5	QtGui::lambda()>::operator()(void) const	spl.cpp	20981								
6	std::function<int ()>::operator()() const	std_f...	302								
7	std::function<int ()>::operator()() const	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()::<lambda()>>(std::_invoke_o...	invok...	60								

Type to locate (Ctrl...) 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 7 Test Results

Sun 12:46 PM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

Welcome to CS106B [master]

- Welcome to CS106B.pro
- Headers
- Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
- Other files

Line: 62, Col: 5

Name	Value
_for_begin	100'd'
_for_end	0'\000'
_for_range	"AdaLovelace"
ch	'd' 100
first	"Ada"
hashVal	???
kLargePrime	16908799
kSmallPrime	127
prime	"Lovelace"

47 * but we thought it might be fun!)

48 */

49 int nameHash(string first, string last){

50 /* This hashing scheme needs two prime numbers, a large prime and a small

51 * prime. These numbers were chosen because their product is less than

52 * $2^{31} - kLargePrime - 1$.

53 */

54 static const int kLargePrime = 16908799;

55 static const int kSmallPrime = 127;

56

57 int hashVal = 0;

58

59 /* Iterate across all the characters in the first name, then the last

60 * name, updating the hash at each step.

61 */

62 for (char ch : first){

63 }

64 }

65 }

66 }

67 }

68 }

69 }

70 }

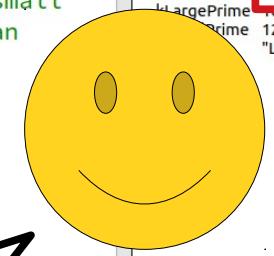
71 }

Now, take a look back at these buttons.

Debugger GDB for "Welcome to CS106B" Threads: #7 Welcome_to_CS10 ... Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	62								
2	qMain	Nam...	31								
3	std::function<int (), int (*)()>::_M_invoke(std::Any_data const&)	std_f...	302								
4	std::function<int ()>::operator()()	std_f...	706								
5	QGui::lambda()::operator()(void) const	spl.cpp	20981								
6	std::function<int (), QtGui::startBackgroundEventLoop(GThunkin...	std_f...	302								
7	std::function<int ()>::operator()()	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_<void, GThreadStd::start()::<lambda()>>(std::_invoke_o...	invok...	60								

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results



Hover your mouse over the one that's on the far right. When you hover over it, it should say "Step Out."

```
47 * but we thought it might be fun!)
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the
60     * name, updating the hash at each step.
61     */
62     for (char c : first)
63     {
64         hashVal = (hashVal * kSmallPrime + c) % kLargePrime;
65     }
66     for (char c : last)
67     {
68         hashVal = (hashVal * kSmallPrime + c) % kLargePrime;
69     }
70 }
71 }
```

Hover your mouse over the one that's on the far right. When you hover over it, it should say "step Out."

Name	Value	Type
------	-------	------

Level	Function	File	Line	Number Function File
1	nameHash	Nam...	62	
2	qMain	Nam...	31	
3	std::Function_handler<int (), int (*)()>::_M_invoke(std::Any_data const&)	std_f...	302	
4	std::function<int ()>::operator()() const	std_f...	706	
5	QtGui::<lambda()>::operator()() const	spl.cpp	20981	
6	std::Function_handler<int (), QtGui::startBackgroundEventLoop(GThunkInfo const&)>::_M_invoke(std::Any_data const&)	std_f...	302	
7	std::function<int ()>::operator()() const	std_f...	706	
8	GThreadStd::run	spl.cpp	22491	
9	GThreadStd::<lambda()>::operator()() const	spl.cpp	22514	
10	std::__invoke_impl<void GThreadStd::start()::<lambda()>>(std::__invoke_o...	invok...	60	

Type to locate (Ctrl... Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

Sun 12:46 PM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

Welcome to CS106B [master]

- Welcome to CS106B.pro
- Headers
- Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
- Other files

Line: 62, Col: 5

Name Value

_for_begin	100 'd'
_for_end	0 '\000'
_for_range	"AdaLovelace"
ch	'd' 100
first	"Ada"
hashVal	???
kLargePrime	16908799
kSmallPrime	127
prime	"Lovelace"

47 * but we thought it might be fun!)

48 */

49 int nameHash(string first, string last){

50 /* This hashing scheme needs two prime numbers, a large prime and a small

51 * prime. These numbers were chosen because their product is less than

52 * $2^{31} - kLargePrime - 1$.

53 */

54 static const int kLargePrime = 16908799;

55 static const int kSmallPrime = 127;

56

57 int hashVal = 0;

58

59 /* Iterate across all the characters in the first name, then the

60 * name, updating the hash at each step.

61 */

62 for (char ch : first){

63 }

64 }

65 }

66 }

67 }

68 }

69 }

70 }

71 }

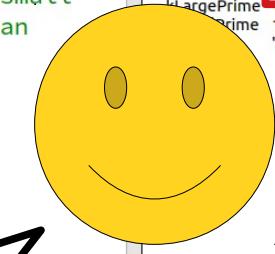
If you click this button, it will keep running this function up until it completes and returns.

Debugger GDB for "Welcome to CS106B" Threads: #7 Welcome_to_CS10 Stopped: "end-stepping-range".

Level Function File Line Number Function File Line Address Condition Ignore Threads

1	nameHash	Nam...	62					
2	qMain	Nam...	31					
3	std::Function<int ()>::_M_invoke(std::Any_data const&)	std_f...	302					
4	std::function<int ()>::operator()()	std_f...	706					
5	QGui::lambda()::operator()(void) const	spl.cpp	20981					
6	std::Function<int ()>::operator()()	std_f...	302					
7	std::function<int ()>::operator()()	std_f...	706					
8	GThreadStd::run	spl.cpp	22491					
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514					
10	std::_invoke_impl<void, GThreadStd::start()::<lambda()>>(std::_invoke_o...	invok...	60					

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results



Sun 12:46 PM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

Line: 62, Col: 5

Name	Value
_for_begin	100'd'
_for_end	0'\000'
_for_range	"AdaLovelace"
ch	'd' 100
first	"Ada"
hashVal	???
kLargePrime	16908799
kSmallPrime	127
prime	"Lovelace"

47 * but we thought it might be fun!)

48 */

49 int nameHash(string first, string last){

50 /* This hashing scheme needs two prime numbers, a large prime and a small

51 * prime. These numbers were chosen because their product is less than

52 * $2^{31} - kLargePrime - 1$.

53 */

54 static const int kLargePrime = 16908799;

55 static const int kSmallPrime = 127;

56

57 int hashVal = 0;

58

59 /* Iterate across all the characters in the first name, then the

60 * name, updating the hash at each step.

61 */

62 for (char ch : first){

63 }

64

65

66

67 }

68 }

69 return hashVal;

70 }

71 }

Now, go click that button. If you did everything right...

Debugger GDB for "Welcome to CS106B" Threads: #7 Welcome_to_CS10 ... Stopped: "end-stepping-range".

Level Function File Line Number Function File Line Address Condition Ignore Threads

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash	Nam...	62								
2	qMain	Nam...	31								
3	std::function<int ()>::_M_invoke(std::any_data const&)	std_f...	302								
4	std::function<int ()>::operator()()	std_f...	706								
5	QGui::lambda()::operator()(void) const	spl.cpp	20981								
6	std::function<int ()>::operator()()	std_f...	302								
7	std::function<int ()>::operator()()	std_f...	706								
8	GThreadStd::run	spl.cpp	22491								
9	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
10	std::_invoke_impl<void, GThreadStd::start()::<lambda()>>(std::_invoke_o...)	invok...	60								

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results



Sun 12:48 PM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

- Welcome to CS106B [master]
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

Name Hash

Name	Value	Type
first	"Ada"	std::string
hashValue	21845	int
last	"Lovelace"	std::string

Line: 31, Col: 5

```
#include "simpio.h" // for getLine
using namespace std;

/* Prototype for the nameHash function. This lets us use the function
 * in main and then define it later in the program.
 */
int nameHash(string first, string last);

int main() {
    string first = getLine("What is your first name? ");
    string last = getLine("What is your last name? ");

    int hashValue = nameHash(first, last);

    cout << "The hash of your name is: " << hashValue << endl;
    return 0;
}

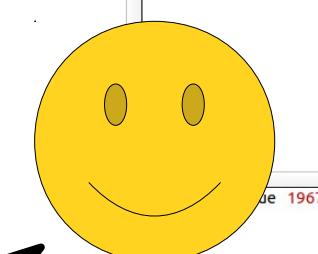
/* This is the actual function that computes the hash code.
 * To talk more about how to do this, we'll have to
 * talk more about what a hash function does.
 * For this assignment, you will implement a simple
 * hash function that takes two strings and returns
 * a single integer value.
 */

... you should end up with something that
looks like this!
```

Debugger GDB for "Welcome to CS106B"

Level	Function
1	qMain
2	std::function<int ()>::_M_invoke(std::any const&)
3	std::function<int ()>::operator()() const
4	QtGui::lambda()::operator()() const
5	std::function<int ()>::operator()() const
6	std::function<int ()>::operator()() const
7	GThreadStd::run
8	GThreadStd::lambda()::operator()() const
9	std::__invoke_impl<void, GThreadStd::start_t::<lambda()>(<lambda()>)(std::__invoke_0...), invok... 60>, invok... 95>
10	std::__invoke<GThreadStd::start_t::<lambda()>(<lambda()>)(GThreadStd::lambda() &), invok... 95>

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results



Sun 12:48 PM
NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

- Welcome to CS106B [master]
 - >Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

Name Hash

Name	Value	Type
first	"Ada"	std::string
hashValue	21845	int
last	"Lovelace"	std::string

Line: 31, Col: 5

```
#include "simpio.h" // for getLine
using namespace std;

/* Prototype for the nameHash function. This lets us use the function
 * in main and then define it later in the program.
 */
int nameHash(string first, string last);

int main() {
    string first = getLine("What is your first name? ");
    string last = getLine("What is your last name? ");

    int hashValue = nameHash(first, last);

    cout << "The hash of your name is: " << hashValue << endl;
    return 0;
}

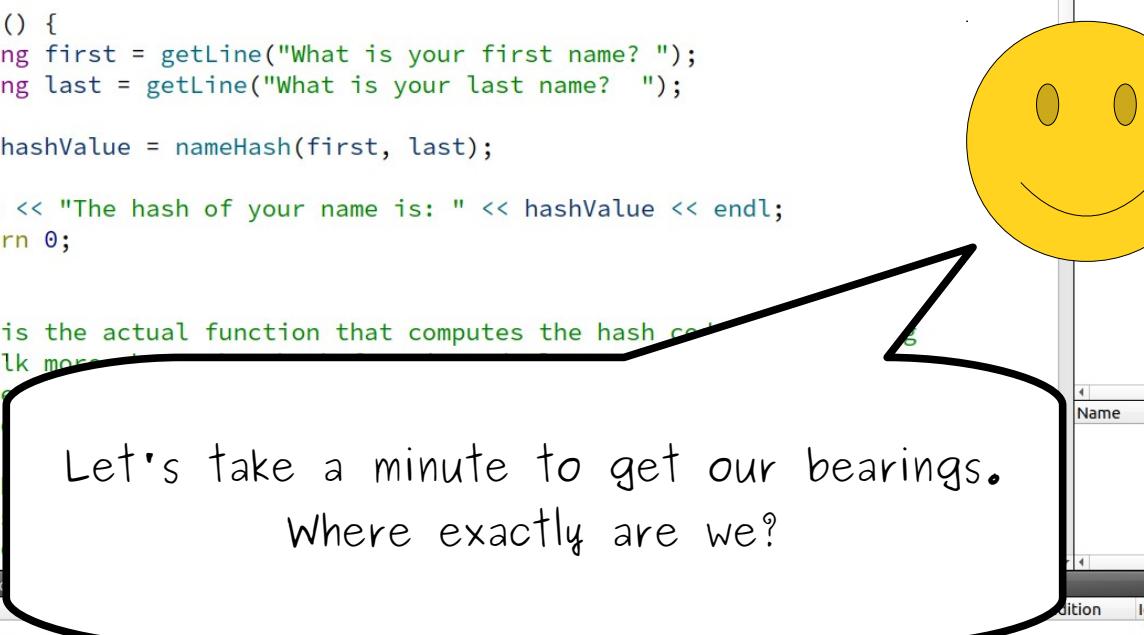
/* This is the actual function that computes the hash code.
 * To talk more about how it works, we'll have to
 * go into more detail.
 * For today, we'll just treat it as a black box.
 * It takes two strings and returns an integer.
 */
```

Debugger GDB for "Welcome to CS106B"

Level	Function
1	qMain
2	std::function<int ()>::_M_invoke(std::any const&)
3	std::function<int ()>::operator()() const
4	QtGui::lambda()::operator()() const
5	std::function<int ()>::operator()() const
6	std::function<int ()>::operator()() const
7	GThreadStd::run
8	GThreadStd::lambda()::operator()() const
9	std::__invoke_impl<void, GThreadStd::start()::lambda()>(<lambda>)(std::__invoke_0...)
10	std::__invoke<GThreadStd::start()::lambda()>(<lambda>)(GThreadStd::lambda() &)

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

Let's take a minute to get our bearings.
Where exactly are we?



Sun 12:48 PM
NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

- Welcome to CS106B [master]
 - >Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

Name Hash Value Type

Name	Value	Type
first	"Ada"	std::string
hashValue	21845	int
last	"Lovelace"	std::string

Line: 31, Col: 5

```
#include "simpio.h" // for getLine
using namespace std;

/* Prototype for the nameHash function. This lets us use the function
 * in main and then define it later in the program.
 */
int nameHash(string first, string last);

int main() {
    string first = getLine("What is your first name? ");
    string last = getLine("What is your last name? ");

    int hashValue = nameHash(first, last);

    cout << "The hash of your name is: " << hashValue << endl;
    return 0;
}

/* This is the actual function that computes the hash code.
 * To talk more about how it works, we'll have to
 * go back to the main function.
 * For the moment, let's just say that
 * treat each character in the string as
 * It the same.
 */


```

Well, the yellow arrow indicates that we're back in `main` again. Cool!

Debugger GDB for "Welcome to CS106B"
Level Function
1 qMain
2 std::function<int ()>::_M_invoke(std::any const&)
3 std::function<int ()>::operator()() const
4 QtGui::lambda()::operator()() const
5 std::function<int ()>::operator()() const
6 std::function<int ()>::operator()() const
7 GThreadStd::run
8 GThreadStd::lambda()::operator()() const
9 std::__invoke_impl<void, GThreadStd::start()::lambda()>(std::__invoke_0... invok...
10 std::__invoke<GThreadStd::start()::lambda()>(GThreadStd::lambda() &) invok...

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

Sun 12:48 PM
NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects Welcome to CS106B [master] Sources lib StanfordCPPLib addr2line.exe addr2line64.exe iconstrip.png splicon-large.png res SRC NameHash.cpp Other files

#include "simpio.h" // for getLine
using namespace std;

/* Prototype for the nameHash function. This lets us use the function
 * in main and then define it later in the program.
 */
int nameHash(string first, string last);

int main()
{
 string first = getLine("What is your first name? ");
 string last = getLine("What is your last name? ");

 int hashValue = nameHash(first, last);

 cout << "The hash of your name is: " << hashValue << endl;
 return 0;

/* This is the actual definition that computes the hash code. We're going
 * to talk more about this in the next section.
 * of the
 *
 * For t
 * treat
 * It th

Please note: This information in the return value panel may not appear on all systems. If you don't see this value, then just move on to the next step of the tutorial.

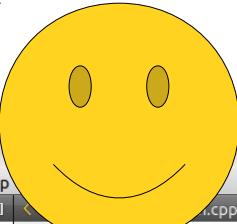
returned value 1967457

We can see that the nameHash function returned 1967457. Thanks, debugger!

Debugger GDB for "Welcome to CS106B"
Level Function
1 qMain
2 std::function<int ()>::__M_invoke(std::any const&) const
3 std::function<int ()>::operator()() const
4 QtGui::lambda()::operator()(void) const
5 std::function<int ()>::operator()() const
6 std::function<int ()>::operator()() const
7 GThreadStd::run
8 GThreadStd::<lambda()>::operator()(void) const
9 std::__invoke_impl<void, GThreadStd::start()::<lambda()>(<lambda()>)(std::__invoke_0...), invok... 60
10 std::__invoke<GThreadStd::start()::<lambda()>(<lambda()>)(GThreadStd::<lambda()> &), invok... 95

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

Activities Qt Creator ▾



Sun 12:48 PM
NameHash.cpp (src @ Welcome to CS106B [master] - Qt Creator)

```
File Edit Build Debug Analyze Tools Window Help
Projects Welcome to CS106B [master]
  Sources
    lib
      StanfordCPPLib
        spl.cpp
        addr2line.exe
        addr2line64.exe
        iconstrip.png
        splicon-large.png
    res
    SRC
      NameHash.cpp
  Other files
```

19 #include "simpio.h" // for getLine
20 using namespace std;
21
22 /* Prototype
 * in main area
 */
23 int nameHash(
24 string name,
25 string hashValue);
26
27 int main()
28 {
29 string name;
30 string hashValue;
31
32 cout << "Please enter your name: " << endl;
33 getline(cin, name);
34 cout << "The hash of your name is: " << hashValue << endl;
35 return 0;
36 }
37 /* This is the actual function.
 * To talk more about what it does
 * in the meantime, think of it as
 * a polynomial over the finite field
 * of the input and produce a
 * number between 0 and 128.
 * For those of you who are more mathematically inclined, this function
 * treats each character in the input name as a number between 0 and 128.
 * It then uses them as coefficients in a polynomial over the finite field

Debugger GDB for "Welcome to CS106B" Threads: #7 Welcome_to_CS10 Stopped: "function-finished".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	qMain	Nam...	31								
2	std::function<int ()>::_M_invoke(std::any const&)	std_f...	302								
3	std::function<int ()>::operator()() const	std_f...	706								
4	QtGui::lambda()::operator()() const	spl.cpp	20981								
5	std::function<int ()>::operator()() const	std_f...	302								
6	std::function<int ()>::operator()() const	std_f...	706								
7	GThreadStd::run	spl.cpp	22491								
8	GThreadStd::lambda()::operator()() const	spl.cpp	22514								
9	std::__invoke_impl<void, GThreadStd::start_t::lambda()>::operator()()	invok...	60								
10	std::__invoke<GThreadStd::start_t::lambda()>::operator()(&)	invok...	95								

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

But if you look up over here in the values window, you can see that `hashValue` has some really weird-looking number stored in it. (You'll almost certainly see something different on your system.)

"Really weird-looking number" is a relative term here, as the number will vary from computer to computer and may be as innocent looking as "0" or "1" or as wild as "2349980890".



Sun 12:48 PM NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects Welcome to CS106B [master]

- >Welcome
- Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp

Other files

```
19 #include "simpio.h" //  
20 using namespace std;  
21  
22 /* Prototype  
   * in main area  
   */  
23 int nameHash(string first, string last);  
24  
25 int main()  
26 {  
27     string first, last;  
28     cout << "Please enter your first name: " << endl;  
29     cin >> first;  
30     cout << "Please enter your last name: " << endl;  
31     int hashValue = nameHash(first, last);  
32  
33     cout << "The hash of your name is: " << hashValue << endl;  
34     return 0;  
35 }  
36  
37 /* This is the actual function that computes the hash code. We're going  
   * to talk more about what hash functions do later in the quarter. In  
   * the meantime, think of it as a function that scrambles up the characters  
   * of the input and produces a number.  
   */  
38  
39 /* For those of you who are more mathematically inclined, this function  
   * treats each character in the input name as a number between 0 and 128.  
   * It then uses them as coefficients in a polynomial over the finite field
```

Debugger GDB for "Welcome to CS106B" Threads: #7 Welcome_to_CS10 Stopped: "function-finished".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	qMain	Nam...	31								
2	std::Function_handler<int (), int (*)()>::_M_invoke(std::Any_data const&)	std_f...	302								
3	std::function<int ()>::operator()() const	std_f...	706								
4	QtGui::lambda()::operator()() const	spl.cpp	20981								
5	std::Function_handler<int (), QtGui::startBackgroundEventLoop(GThunkInfo const&)>::_M_invoke(std::Any_data const&)	std_f...	302								
6	std::function<int ()>::operator()() const	std_f...	706								
7	GThreadStd::run	spl.cpp	22491								
8	GThreadStd::lambda()::operator()() const	spl.cpp	22514								
9	std::__invoke_impl<void, GThreadStd::start()::lambda()>(<lambda()>)(std::__invoke_0...)	invok...	60								
10	std::__invoke<GThreadStd::start()::lambda()>(GThreadStd::lambda() &&)	invok...	95								

Type to locate (Ctrl...) 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 7 Test Results

But it looks like we're setting `hashValue` equal to the number that was returned by the `nameHash` function. What's going on?

returned value 1967457 int



Sun 12:48 PM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

- Welcome to CS106B [master]
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

Line: 31, Col: 5

Name	Value	Type
first	"Ada"	std::string
hashValue	21845	int
Lovelace	"Lovelace"	std::string

This is pretty cool, actually!

```
#include "simpio.h" //  
using namespace std;  
  
/* Prototype  
 * in main area  
 */  
int nameHash(string first, string last);  
  
int main()  
{  
    string first = getLine("What is your first name? ");  
    string last = getLine("What is your last name? ");  
  
    int hashValue = nameHash(first, last);  
  
    cout << "The hash of your name is: " << hashValue << endl;  
    return 0;  
}  
  
/* This is the actual function that computes the hash code. We're going  
 * to talk more about what hash functions do later in the quarter. In  
 * the meantime, think of it as a function that scrambles up the characters  
 * of the input and produces a number.  
 *  
 * For those of you who are more mathematically inclined, this function  
 * treats each character in the input name as a number between 0 and 128.  
 * It then uses them as coefficients in a polynomial over the finite field
```

Debugger GDB for "Welcome to CS106B" Threads: #7 Welcome_to_CS10 Stopped: "function-finished".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	qMain	Nam...	31								
2	std::Function_handler<int (), int (*)()>::_M_invoke(std::Any_data const&)	std_f...	302								
3	std::function<int ()>::operator()() const	std_f...	706								
4	QtGui::lambda()::operator()(void) const	spl.cpp	20981								
5	std::Function_handler<int (), QtGui::startBackgroundEventLoop(GThunkInfo const&)>::_M_invoke(std::Any_data const&)	std_f...	302								
6	std::function<int ()>::operator()() const	std_f...	706								
7	GThreadStd::run	spl.cpp	22491								
8	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
9	std::__invoke_impl<void, GThreadStd::start()::<lambda()>(&std::__invoke_0...), void, void>::operator()	invok...	60								
10	std::__invoke<GThreadStd::start()::<lambda()>(>hreadStd::lambda() &&), void, void>::operator()	invok...	95								

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results



Sun 12:48 PM

NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

- Welcome to CS106B [master]
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

Line: 31, Col: 5

Name	Value	Type
first	Ada	std::string
hashValue	21845	int
last	"Lovelace"	std::string

What's happened is that we've just returned from nameHash with a value, but since we're going through the program one step at a time, we haven't actually assigned that value to hashValue yet!

```
#include "simpio.h"
using namespace std;

/* Prototype
 * in main and nameHash
 */
int nameHash(string first, string last);

int main()
{
    string first;
    string last;
    int hashValue = nameHash(first, last);

    cout << "The hash of your name is: " << hashValue << endl;
    return 0;
}

/* This is the actual function that computes the hash code. We're going
 * to talk more about what hash functions do later in the quarter. In
 * the meantime, think of it as a function that scrambles up the characters
 * of the input and produces a number.
 *
 * For those of you who are more mathematically inclined, this function
 * treats each character in the input name as a number between 0 and 128.
 * It then uses them as coefficients in a polynomial over the finite field
 */

Debugger GDB for "Welcome to CS106B"
Threads: #7 Welcome_to_CS10 Stopped: "function-finished".
Level Function File Line
1 qMain Nam... 31
2 std::Function_handler<int (), int (*)()>::_M_invoke(std::Any_data const&) std_f... 302
3 std::function<int ()>::operator()() const std_f... 706
4 QtGui::lambda()::operator()(void) const spl.cpp 20981
5 std::Function_handler<int (), QtGui::startBackgroundEventLoop(GThunkin... std_f... 302
6 std::function<int ()>::operator()() const std_f... 706
7 GThreadStd::run spl.cpp 22491
8 GThreadStd::<lambda()>::operator()(void) const spl.cpp 22514
9 std::__invoke_impl<void, GThreadStd::start()::<lambda()>(<lambda()>)(std::__invoke_o... invok... 60
10 std::__invoke<GThreadStd::start()::<lambda()>(<lambda()>)(GThreadStd::<lambda()> &&) invok... 95
```

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results



Sun 12:48 PM NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

- Welcome to CS106B [master]
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

Line: 31, Col: 5

Name	Value	Type
first	"Ada"	std::string
hashValue	21845	int
Lovelace	"Lovelace"	std::string

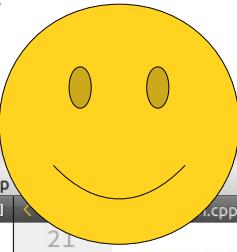
Let's do a "Step Over" so that we can finish executing this line. Click "Step Over," and if you did everything right...

```
19 #include "simpio.h" //  
20 using namespace std;  
21  
22 /* Prototype  
 * in main area  
 */  
23 int nameHash(string first, string last);  
24  
25 int main() {  
26     string first = getLine("What is your first name? ");  
27     string last = getLine("What is your last name? ");  
28  
29     int hashValue = nameHash(first, last);  
30  
31     cout << "The hash of your name is: " << hashValue << endl;  
32  
33     return 0;  
34 }  
35  
36  
37 /* This is the actual function that computes the hash code. We're going  
 * to talk more about what hash functions do later in the quarter. In  
 * the meantime, think of it as a function that scrambles up the characters  
 * of the input and produces a number.  
 *  
 * For those of you who are more mathematically inclined, this function  
 * treats each character in the input name as a number between 0 and 128.  
 * It then uses them as coefficients in a polynomial over the finite field
```

Debugger GDB for "Welcome to CS106B"  Threads: #7 Welcome_to_CS10 Stopped: "function-finished".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	qMain	Nam...	31								
2	std::function<int ()>::_M_invoke(std::any_data const&)	std_f...	302								
3	std::function<int ()>::operator()()	std_f...	706								
4	QtGui::lambda()::operator()()	splcpp	20981								
5	std::function<int ()>::operator()()	std_f...	302								
6	std::function<int ()>::operator()()	std_f...	706								
7	GThreadStd::run	splcpp	22491								
8	GThreadStd::lambda()::operator()()	splcpp	22514								
9	std::__invoke_impl<void, GThreadStd::start_t::lambda()>(&std::__invoke_0...)	invok...	60								
10	std::__invoke<GThreadStd::start_t::lambda()>(GThreadStd::lambda() &&)	invok...	95								

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results



Sun 12:52 PM
NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

- Welcome to CS106B [master]
 - Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - silicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

```
21  /* Prototype for the
22   * function we're going to call.
23   */
24   */
25 int nameHash(string first, string last);
26
27 int main()
28 {
29     string first;
30     string last;
31
32     int hashValue = nameHash(first, last);
33
34     cout << "The hash of your name is: " << hashValue << endl;
35     return 0;
36 }
37
38 /* This is the actual function that computes the hash code. We're going
39 * to talk more about what hash functions do later in the quarter. In
40 * the meantime, think of it as a function that scrambles up the characters
41 * of the input and produces a number.
42 *
43 * For those of you who are more mathematically inclined, this function
44 * treats each character in the input name as a number between 0 and 128.
45 * It then uses them as coefficients in a polynomial over the finite field
46 * F_p, where p is a large prime number, and evaluates that polynomial at
47 * some smaller prime number q. (You aren't expected to know this for CS106B,
```

... you should see the right value get stored
(notice it's in red!) and we've moved to the
next line.

Name	Value	Type
first	"Ada"	std::string
hashValue	1967457	int

Name	Value	Type
last	"Lovelace"	std::string

Debugger GDB for "Welcome to CS106B"

Threads: #7 Welcome_to_CS10 Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	qMain	Nam...	33								
2	std::Function_handler<int (), int (*)()>::_M_invoke(std::Any_data const&)	std_f...	302								
3	std::function<int ()>::operator()()	std_f...	706								
4	QtGui::lambda()::operator()()	spl.cpp	20981								
5	std::Function_handler<int (), QtGui::startBackgroundEventLoop(GThunkin...	std_f...	302								
6	std::function<int ()>::operator()()	std_f...	706								
7	GThreadStd::run	spl.cpp	22491								
8	GThreadStd::<lambda()>::operator()()	spl.cpp	22514								
9	std::__invoke_impl<void, GThreadStd::start()::<lambda()>(<lambda()>)(std::__invoke_o...	invok...	60								
10	std::__invoke<GThreadStd::start()::<lambda()>(<lambda()>)(&&)()	invok...	95								



Sun 12:52 PM NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

- Welcome to CS106B [master]
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

21
22 */* Prototype for nameHash */*
23 *in main.cxx*
24 **/*
25 int nameHash(
26 string first,
27 string last);
28
29 int main()
30 string first,
31 string last;
32
33 cout << "The hash of your name is: " << hashValue << endl;
34 return 0;
35 }
36
37 */* This is the actual function that computes the hash code. We're going*
to talk more about what hash functions do later in the quarter. In
the meantime, think of it as a function that scrambles up the characters
of the input and produces a number.
38 ***
39 *For those of you who are more mathematically inclined, this function*
treats each character in the input name as a number between 0 and 128.
40 *It then uses them as coefficients in a polynomial over the finite field*
41 *F_p, where p is a large prime number, and evaluates that polynomial at*
42 *some smaller prime number q. (You aren't expected to know this for CS106B,*
43 *but it's a cool trick!)*
44
45
46

At this point, we've seen just about everything we care about. Rather than single-stepping all the way to the end, let's just tell the program to keep on running.

Debugger GDB for "Welcome to CS106B" Threads: #7 Welcome_to_CS10 Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	qMain	Nam...	33								
2	std::function<int ()>::_M_invoke(std::any const&)	std_f...	302								
3	std::function<int ()>::operator()() const	std_f...	706								
4	QtGui::lambda()::operator()() const	spl.cpp	20981								
5	std::function<int ()>::operator()() const	std_f...	302								
6	std::function<int ()>::operator()() const	std_f...	706								
7	GThreadStd::run	spl.cpp	22491								
8	GThreadStd::<lambda()>::operator()() const	spl.cpp	22514								
9	std::__invoke_impl<void, GThreadStd::start_t::<lambda()>(&std::__invoke_0...)	invok...	60								
10	std::__invoke<GThreadStd::start_t::<lambda()>(>hreadStd::<lambda()> &&)	invok...	95								

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results

Sun 12:52 PM
NameHash.cpp (src @ Welcome to CS106B) [master] - Qt Creator

File Edit Build Debug Analyze Tools Window Help

Projects

- Welcome to CS106B [master]
 - >Welcome to CS106B.pro
 - Headers
 - Sources
 - lib
 - StanfordCPPLib
 - spl.cpp
 - addr2line.exe
 - addr2line64.exe
 - iconstrip.png
 - splicon-large.png
 - res
 - src
 - NameHash.cpp
 - Other files

Line: 33, Col: 5

Name	Value	Type
first	"Ada"	std::string
hashValue	1967457	int
last	"Lovelace"	std::string

```
21
22  /* Prototype for the nameHash function. This lets us use the function
23   * in main and then define it later in the program.
24   */
25  int nameHash(string first, string last);
26
27  int main() {
28      string first = getLine("What is your first name? ");
29      string last = getLine("What is your last name? ");
30
31      int hashValue = nameHash(first, last);
32
33      cout << "The hash of your name is: " << hashValue << endl;
34      return 0;
35  }
36
37  /* This is the actual function that computes the hash
38   * to talk more about what hash functions do later
39   * the meantime, think of it as a function that scr
40   * of the input and produces a number.
41   *
42   * For those of you who are

```

To do this, click on this button. If you hover over it, it says "Continue," and that button means "unpause the program and let it keep running from here."

Views

Wel...106B

Debug

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML Debugger Console General Messages Test Results





If you do, you should see something like this.
(The program window might not automatically
pop up. That's okay! Just open it manually.)

Our program is now done running!

```
int main()
{
    string first;
    cout << "What is your first name? " << first;
    string last;
    cout << "What is your last name? " << last;
    cout << "The hash of your name is: " << hash(first + last);
}

/* This
 * to ta
 * the m
 * of th
 *
 * For t
 * treat
 * It th
 * F_p,
 * some
```

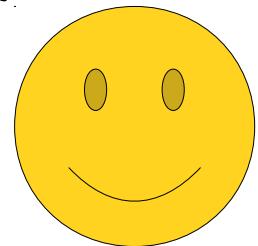
What is your first name? Ada
What is your last name? Lovelace
The hash of your name is: 1967457

Debugger GDB for "Welcome to CS106B" | Thread: #6 Welcome to CS106B | Running.

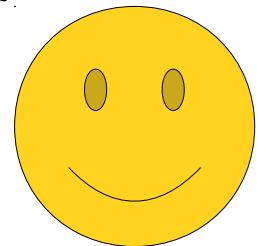
Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
1	qMain	Nam...	33								
2	std::function<int ()>::_M_invoke(std::_Any_data const&)	std_f...	302								
3	std::function<int ()>::operator()() const	std_f...	706								
4	QtGui::lambda()::operator()(void) const	spl.cpp	20981								
5	std::function<int ()>::operator<int(), QtGui::startBackgroundEventLoop(GThunkin...	std_f...	302								
6	std::function<int ()>::operator()() const	std_f...	706								
7	GThreadStd::run	spl.cpp	22491								
8	GThreadStd::<lambda()>::operator()(void) const	spl.cpp	22514								
9	std::__invoke_impl<void, GThreadStd::start()::<lambda()>(std::__invoke_...>, invok...	invok...	60								
10	std::__invoke<GThreadStd::start()::<lambda()>::(GThreadStd::lambda()> &) invok...	invok...	95								

Type to locate (Ctrl...) | Issues | Search Results | Application Output | Compile Output | QML Debugger Console | General Messages | Test Results

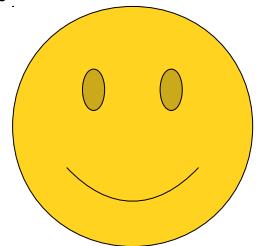
So there you have it! You've now gotten more
familiar with the debugger!



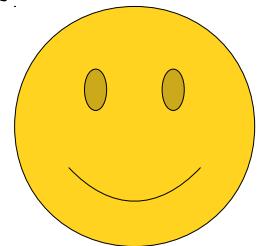
You know how to set a breakpoint to pause the program at a particular point.



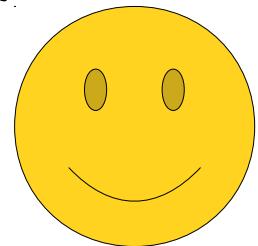
You know how to read the call stack and to see the values of local variables.



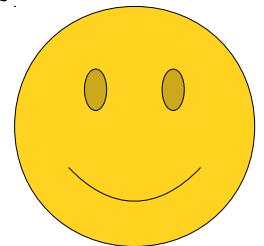
You know how to single-step the program and see what values change.



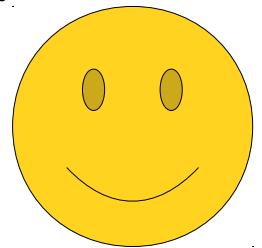
You know how to run a function to completion,
and how to let the program keep on running.



As you write more and more complicated programs this quarter, you'll get a lot more familiar using the debugger and seeing how your programs work.



And, if you continue to build larger and larger pieces of software, you'll find that knowing how to use a debugger is a surprisingly valuable skill!



Hope this helps, and welcome to CS106B!

