/*
 * CS 106B, Chris Gregg
 * This program contains code to demonstrate pointers.
 * NOTE 1: This program will compile with warnings due to uninitialized variables.
 * Note 2: This program will crash (purposefully) on some choices.
 */

#include <iostream>
#include "console.h"
#include "simpio.h"

using namespace std;

// function prototype declarations
int requestTest();
void addressExample();
void crash();
void sameVariablePointers();
void derefExample();
void pointerExample();

int main() {
    int testNum = requestTest();
    while (testNum != -1) {
        switch (testNum) {
            case 0: // address example
                addressExample();
                break;
            case 1: // will crash...
                crash();
                break;
            case 2:
                sameVariablePointers();
                break;
            case 3:
                derefExample();
                break;
            case 4:
                pointerExample();
                break;
        }
        cout << endl;
        testNum = requestTest();
    }
    cout << "Goodbye!" << endl;
    return 0;
}

int requestTest() {
    int response = -2; // initial condition
    cout << "Pointer Examples" << endl;
    cout << "0. Pointers have addresses" << endl;
    cout << "1. Seg Fault!" << endl;
    cout << "2. Pointers to the same variable" << endl;
    cout << "3. Dereference example" << endl;
    cout << "4. Pointer example with crash at end." << endl;
    cout << endl;
    while (response < -1 or response > 4) {
        response = getInteger("Please choose an option (0-2), -1 to quit: ");
    }
    return response;
}

void addressExample() {
    string s;
    int i;
    double sum;

    cout << "Variables uninitialized" << endl
         << "    Address of s  :  " << &s << endl
         << "    Address of i  :  " << &i << endl
         << "    Address of sum:  " << &sum << endl
```cpp
<< "    Value of s    :  "   << s    << endl
<< "    Value of i    :  "   << i    << endl // will cause warning
<< "    Value of sum  :  "   << sum  << endl // will cause warning
<< endl;

s   = "Bazinga";
sum = 0.0;
i   = 2;
cout << "---------------------" << endl;
cout << "Variables initialized" << endl
<< "    Address of s  :  " << &s   << endl
<< "    Address of i  :  " << &i   << endl
<< "    Address of sum:  " << &sum << endl
<< "    Value of s    :  " << s    << endl
<< "    Value of i    :  " << i    << endl
<< "    Value of sum  :  " << sum  << endl
<< endl;
while (i < 1000) {
    sum += i;
i *= 2;
    cout << "---------------------" << endl;
cout << "Ready to do loop test again" << endl
<< "    Address of i  :  " << &i   << endl
<< "    Address of sum:  " << &sum << endl
<< "    Value of i    :  " << i    << endl
<< "    Value of sum  :  " << sum  << endl
<< endl;
}

// This function will attempt to dereference a NULL pointer, which will crash
// the program. :(  
void crash() {
    string *sPtr = NULL;
    string s = "hello";
    cout << *sPtr << endl;
}
void derefExample() {
    string *sPtr = NULL;
    string s = "hello";
sPtr = &s;
    *sPtr = "goodbye";
    cout << *sPtr << endl;
}
void pointerExample() {
    int      x;
    int     *p;  // declaration of a pointer to an int
    int     *q;  // another pointer to an int
    int    **z;  // declaration of a pointer to a pointer to an int

    /**** STAGE 1 *****/
    cout << endl << "stage 1, initialization" << endl;
    x = 3;
p = &x;  //the & operator finds the address of the variable
    cout << "    p: "  <<  p << endl;  // print the address of x
    cout << "    *p: "  << *p << endl;  // print the value of x;
cout << "    x: "  <<  x << endl;  // print the value of x;

    /**** STAGE 2 *****/
cout << endl;
cout << "stage 2, *p = *p - 1 " << endl;
    *p = *p - 1;
cout << "    x: "  <<  x << endl;  // value of x has decreased by 1
    cout << "    *p: "  << *p << endl;  // print the value of x thru p
```
151     /**** STAGE 3 ******/
152     cout << endl;
153     cout << "stage 3, q = p " << endl;
154     q = p;
155
156     cout << " q: " << q << endl;   // should be the same value as p
157     cout << " *q: " << *q << endl;   // which points to x
158
159     /**** STAGE 4 ******/
160     cout << endl;
161     cout << "stage 4, *q = *p - 1" << endl;
162     *q = *p - 1;
163
164     // *p and *q and x should all be the same thing
165     // (which is now x = x - 1)
166     cout << " *p: " <<  *p << endl;
167     cout << " *q: " <<  *q << endl;
168     cout << "  x: " <<  x << endl;
169
170     /**** STAGE 5 ******/
171     cout << endl;
172     cout << "stage 5a, z = &p" << endl;
173     z = &p;
174
175     cout << "  z: " <<   z << endl;
176     cout << " *z: " <<  *z << endl;
177     cout << " **z: " << **z << endl;
178
179     cout << "stage 5a, z = &q" << endl;
180     z = &q;
181
182     cout << "  z: " <<   z << endl;
183     cout << " *z: " <<  *z << endl;
184     cout << " **z: " << **z << endl;
185
186     /**** STAGE 6 ******/
187     cout << endl;
188     cout << "stage 6, p = (int*) 4" << endl;
189     p = (int*) 4;                   // p assigned the location 0x04
190
191     cout << "  p: " << p << endl;
192     cout << "  q: " << q << endl;
193     cout << "The data stored at location 4 is: " ;
194     cout << *p << endl;              // this will cause a seg fault!
195 }
196
197 void sameVariablePointers() {
198     string *sp1 = NULL;
199     string *sp2 = NULL;
200     string s = "hello";
201     sp1 = &s;
202     cout << *sp1 << endl;
203     sp2 = sp1;
204     cout << *sp2 << endl;
205     *sp1 = "goodbye";
206     cout << *sp1 << endl;
207     cout << *sp2 << endl;
208 }