Announcements

Labs happened

C-strings, gdb

assign0 graded

assign1 due Monday

Goals for Today

Explore tradeoffs between stack and heap

See pointers used to pass by reference

See and debug common memory errors

- gdb and Valgrind

Stack vs. Heap Arrays

Stack: `int arr[5];`

- Automatically cleaned up
- “Cheap”: efficient to allocate/deallocate

Heap: `int *p = malloc(5 * sizeof(int));`

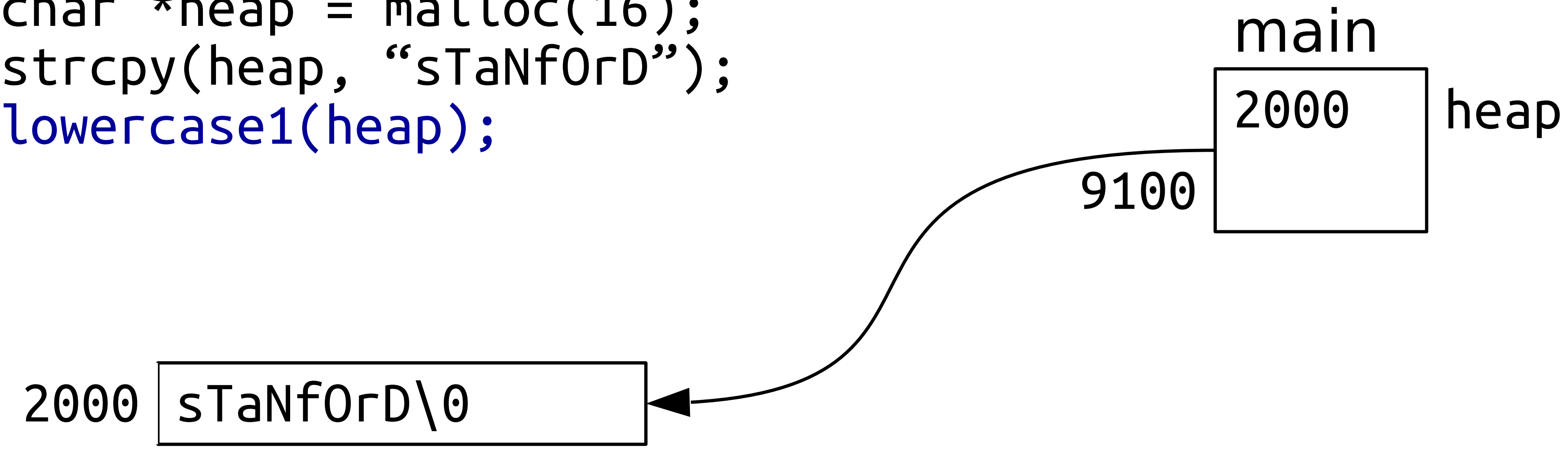
- Precisely control when memory deallocated
- Can be resized (realloc)

Rule of thumb: Use stack when possible, heap when necessary.

Code Examples

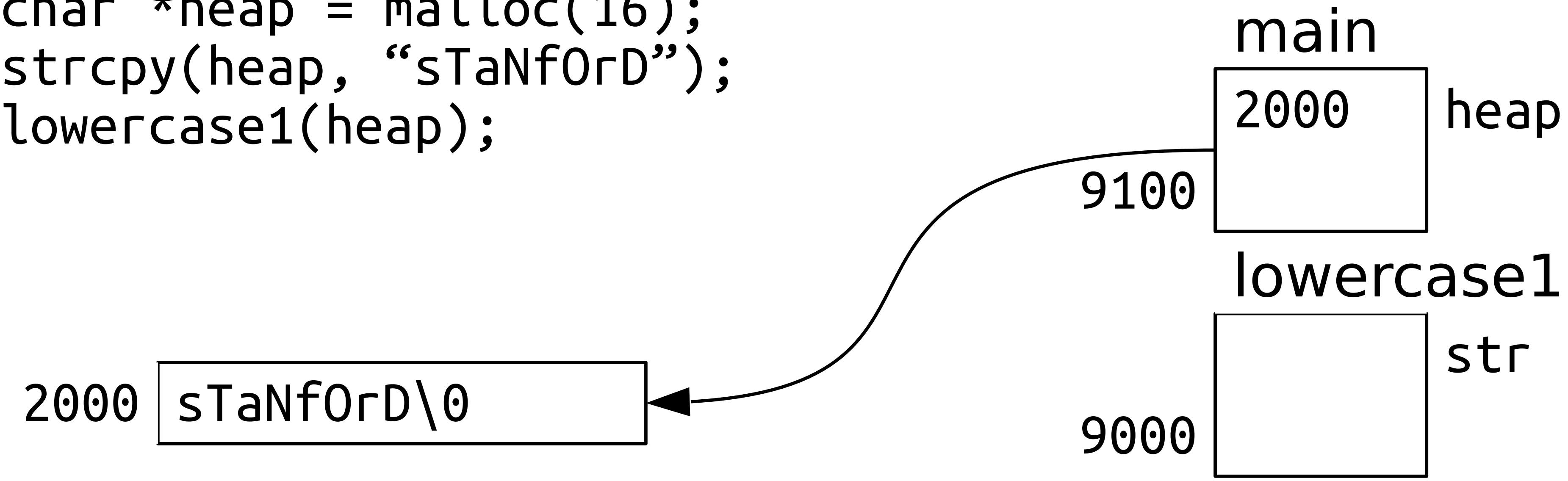
lowercase1

```
void lowercase1(char *str) {
  for (int i = 0; i < strlen(str); i++)
    str[i] = tolower(str[i]);
}

int main() {
  char *heap = malloc(16);
  strcpy(heap, “sTaNfOrD”);
  lowercase1(heap);
}
```



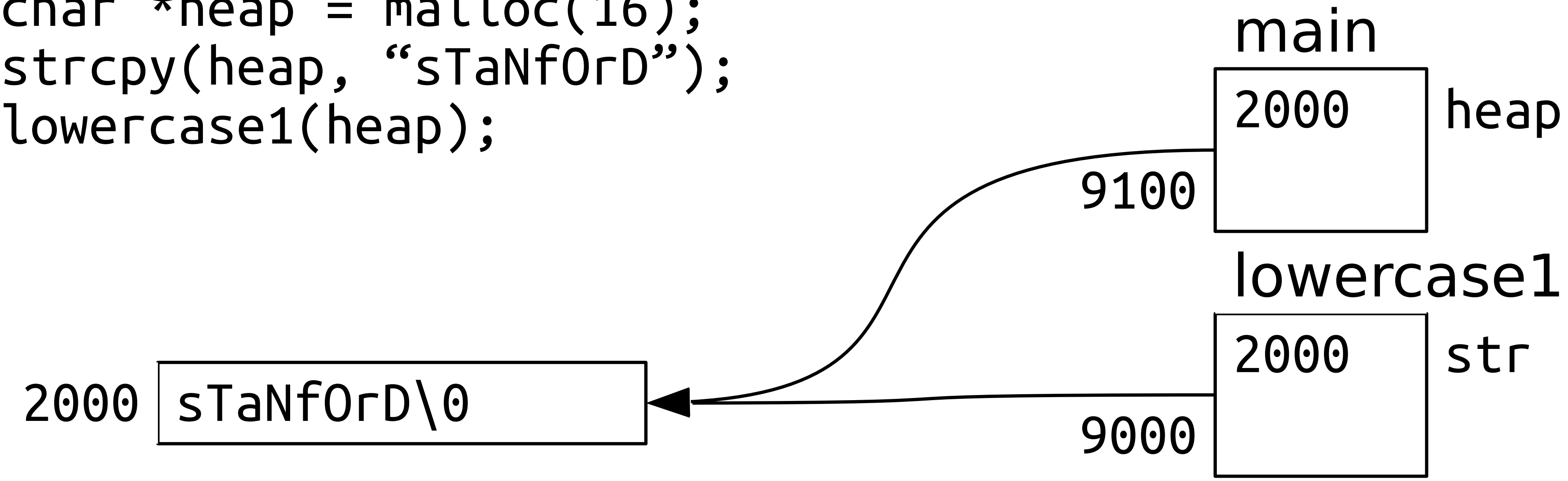
lowercase1

```
void lowercase1(char *str) {
  for (int i = 0; i < strlen(str); i++)
    str[i] = tolower(str[i]);
}

int main() {
  char *heap = malloc(16);
  strcpy(heap, “sTaNfOrD”);
  lowercase1(heap);
}
```



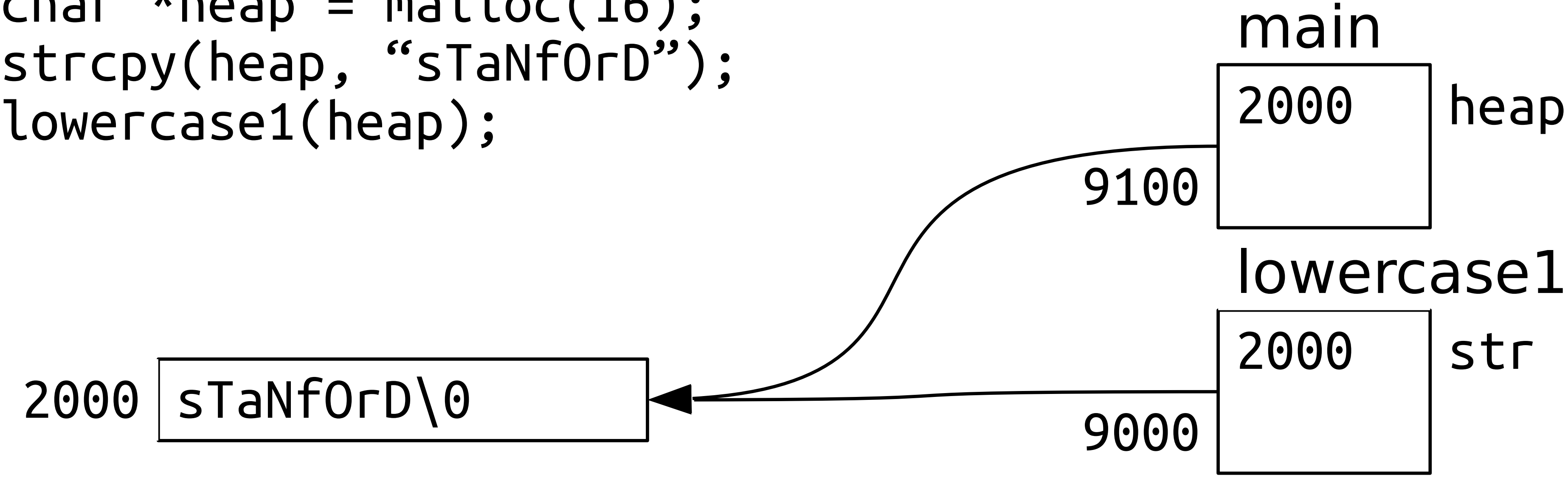
lowercase1

```
void lowercase1(char *str) {
  for (int i = 0; i < strlen(str); i++)
    str[i] = tolower(str[i]);
}

int main() {
  char *heap = malloc(16);
  strcpy(heap, “sTaNfOrD”);
  lowercase1(heap);
}
```



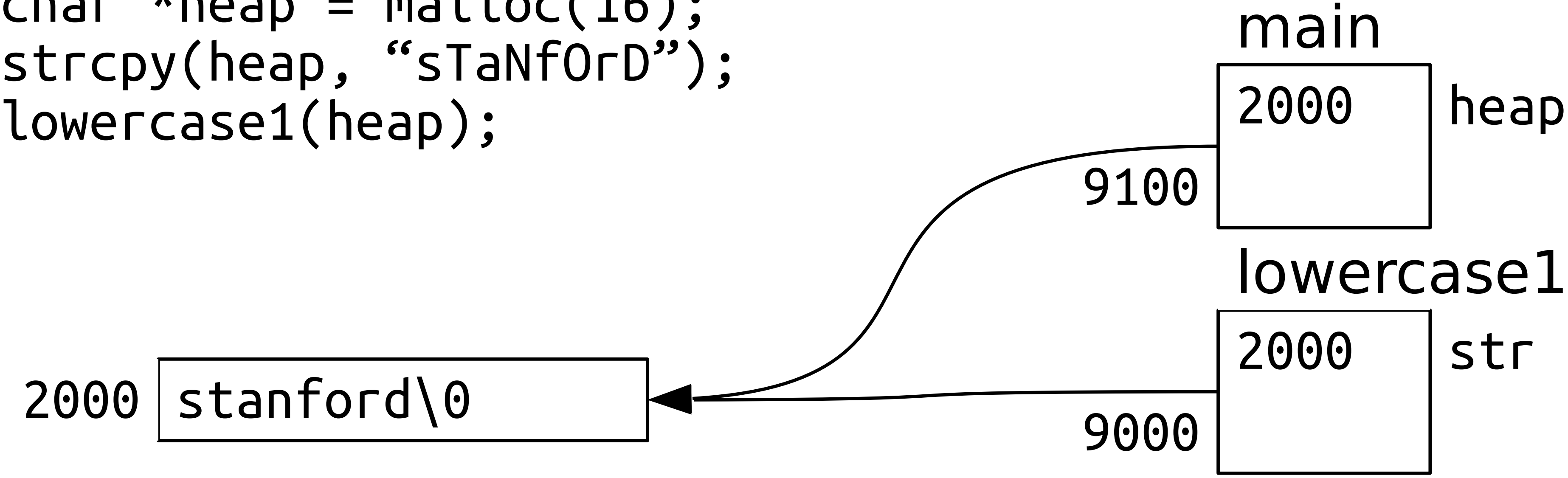
lowercase1

```
void lowercase1(char *str) {
  for (int i = 0; i < strlen(str); i++)
    str[i] = tolower(str[i]);
}

int main() {
  char *heap = malloc(16);
  strcpy(heap, “sTaNfOrD”);
  lowercase1(heap);
}
```



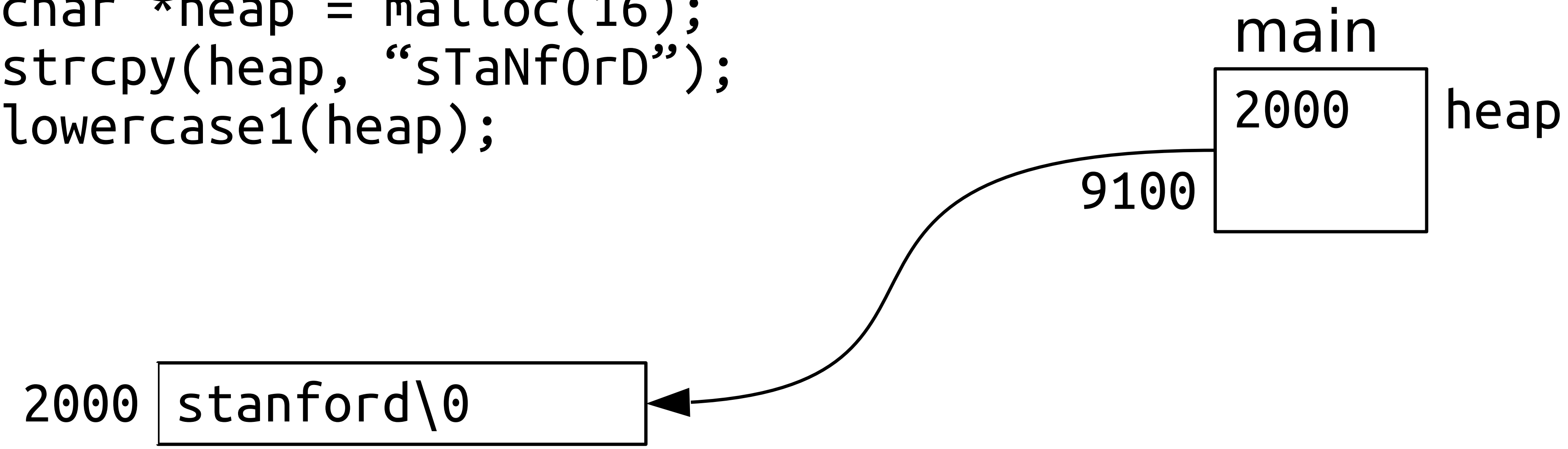
lowercase1

```
void lowercase1(char *str) {
  for (int i = 0; i < strlen(str); i++)
    str[i] = tolower(str[i]);
}

int main() {
  char *heap = malloc(16);
  strcpy(heap, “sTaNfOrD”);
  lowercase1(heap);
}
```



lowercase1

```
void lowercase1(char *str) {
  for (int i = 0; i < strlen(str); i++)
    str[i] = tolower(str[i]);
}

int main() {
  char *heap = malloc(16);
  strcpy(heap, “sTaNfOrD”);
  lowercase1(heap);
}
```



main

2000

heap

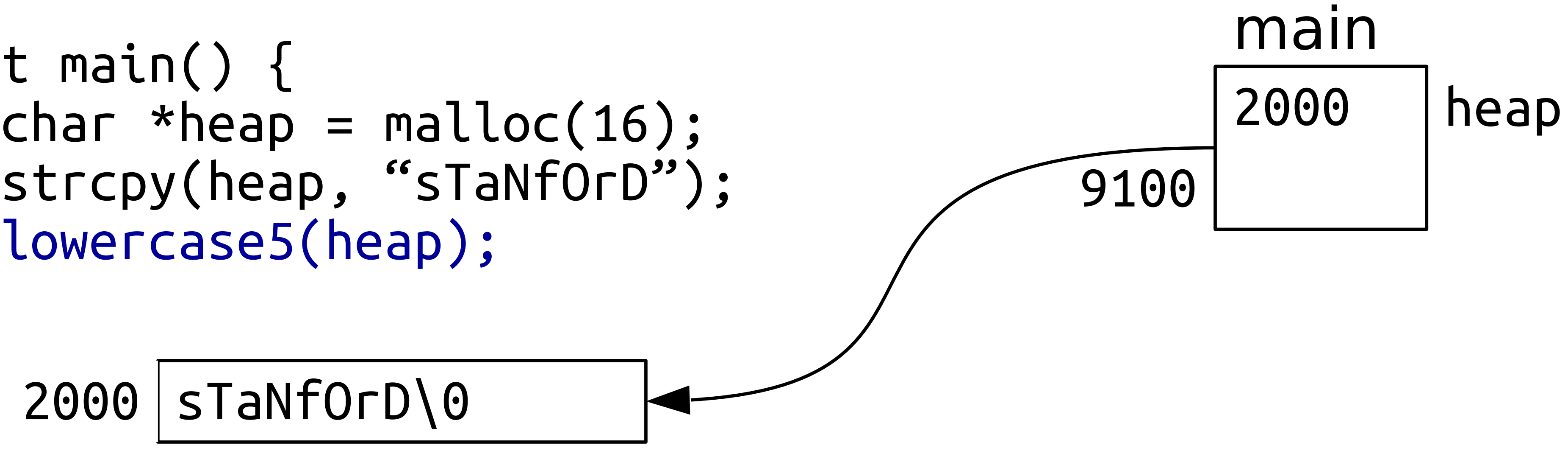
9100

2000

stanford\0

lowercase5

```
void lowercase5(char *str) {
  char *copy = malloc(strlen(str) + 1);
  for (int i = 0; i <= strlen(str); i++)
    copy[i] = tolower(str[i]);
  str = copy;
}

int main() {
  char *heap = malloc(16);
  strcpy(heap, “sTaNfOrD”);
  lowercase5(heap);
}
```



main

2000

heap

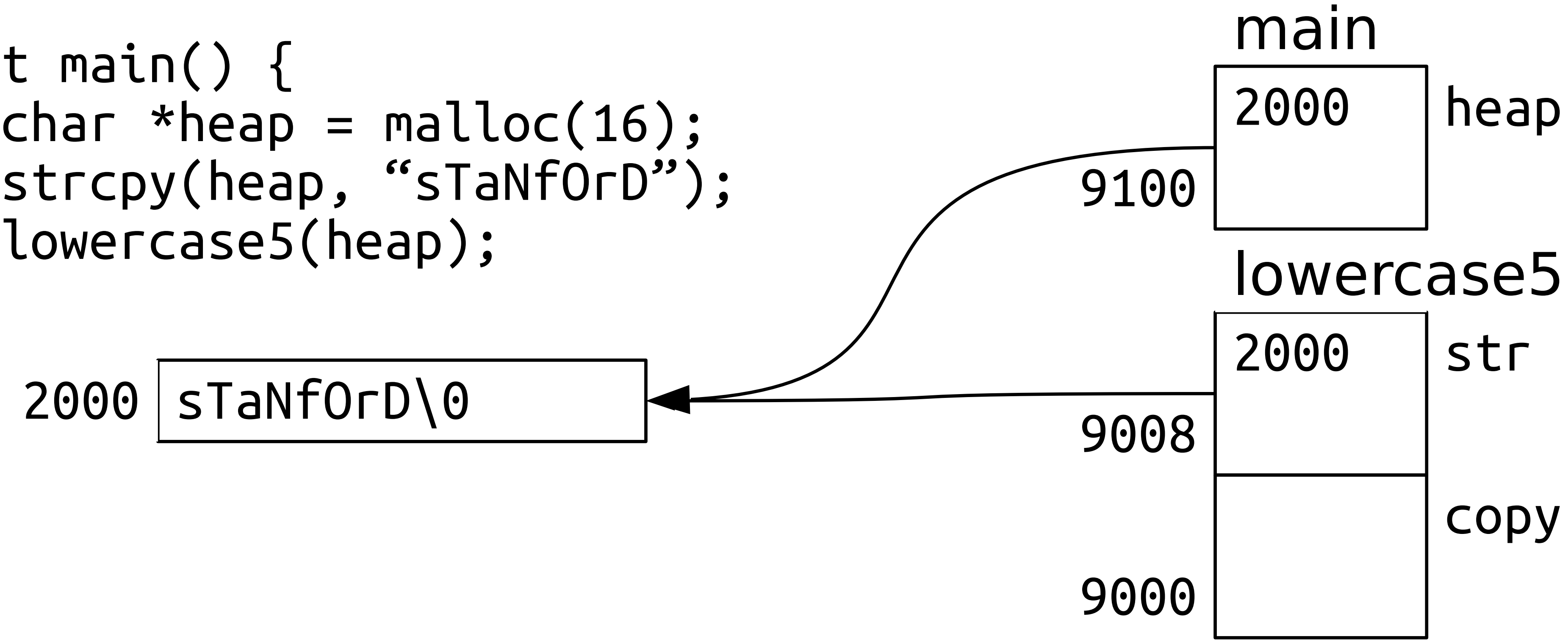
9100

2000

sTaNfOrD\0

lowercase5

```
void lowercase5(char *str) {
  char *copy = malloc(strlen(str) + 1);
  for (int i = 0; i <= strlen(str); i++)
    copy[i] = tolower(str[i]);
  str = copy;
}

int main() {
  char *heap = malloc(16);
  strcpy(heap, “sTaNfOrD”);
  lowercase5(heap);
}
```



main

2000 heap

9100

lowercase5

2000 str

9008

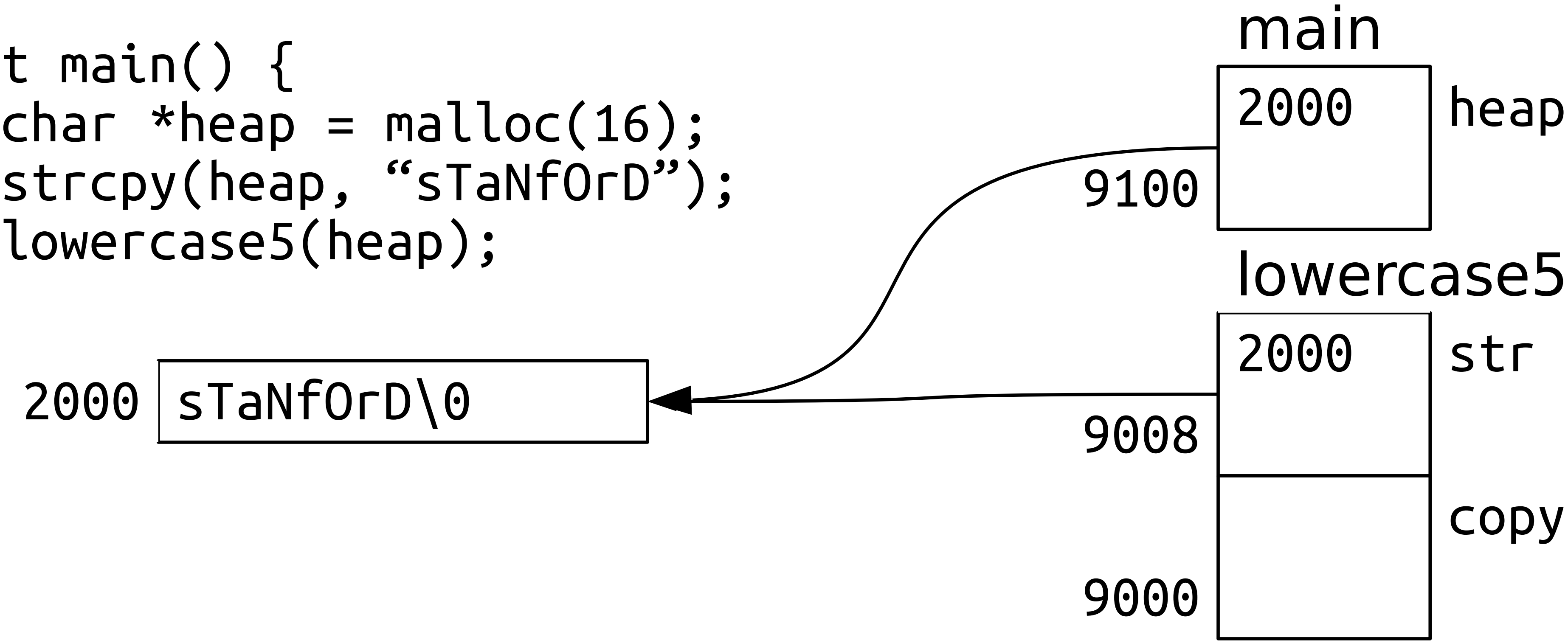
copy

9000

2000 sTaNfOrD\0

lowercase5

```
void lowercase5(char *str) {
  char *copy = malloc(strlen(str) + 1);
  for (int i = 0; i <= strlen(str); i++)
    copy[i] = tolower(str[i]);
  str = copy;
}

int main() {
  char *heap = malloc(16);
  strcpy(heap, “sTaNfOrD”);
  lowercase5(heap);
}
```



main

2000 heap

9100

lowercase5

2000 str

9008

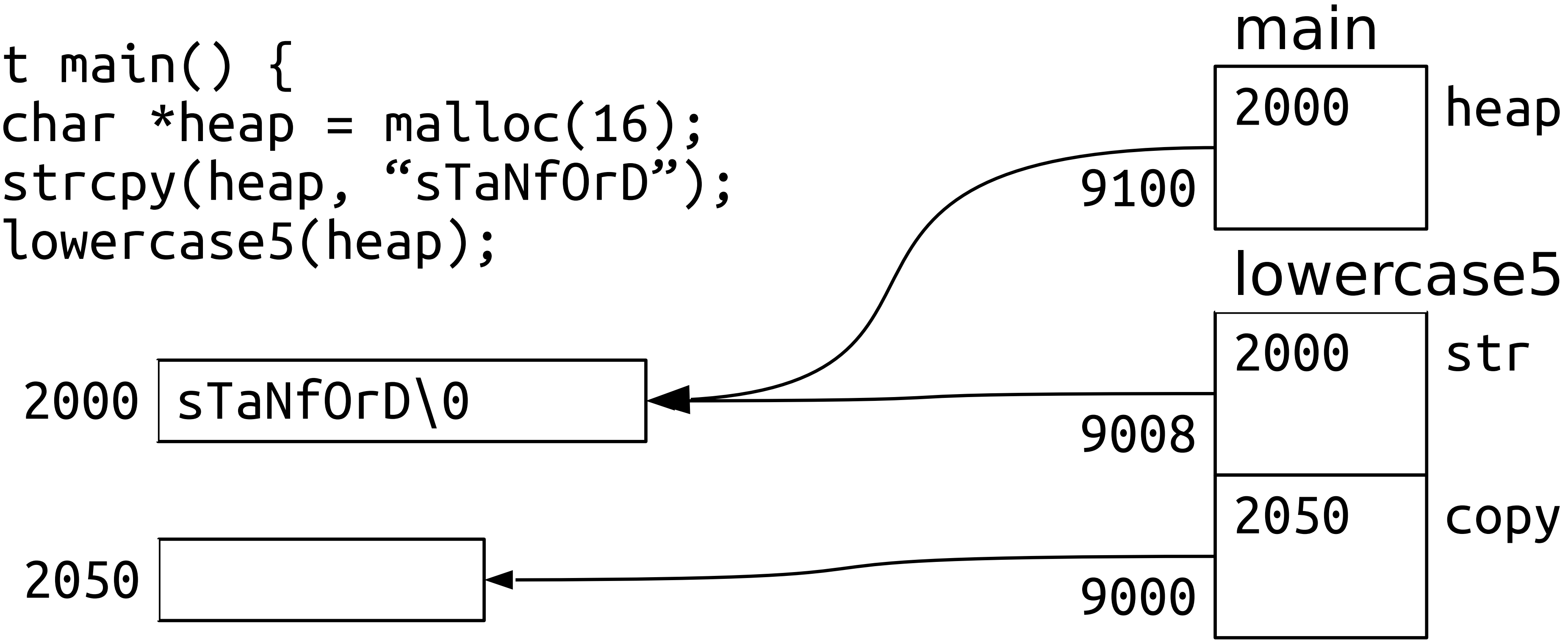
copy

9000

2000 sTaNfOrD\0

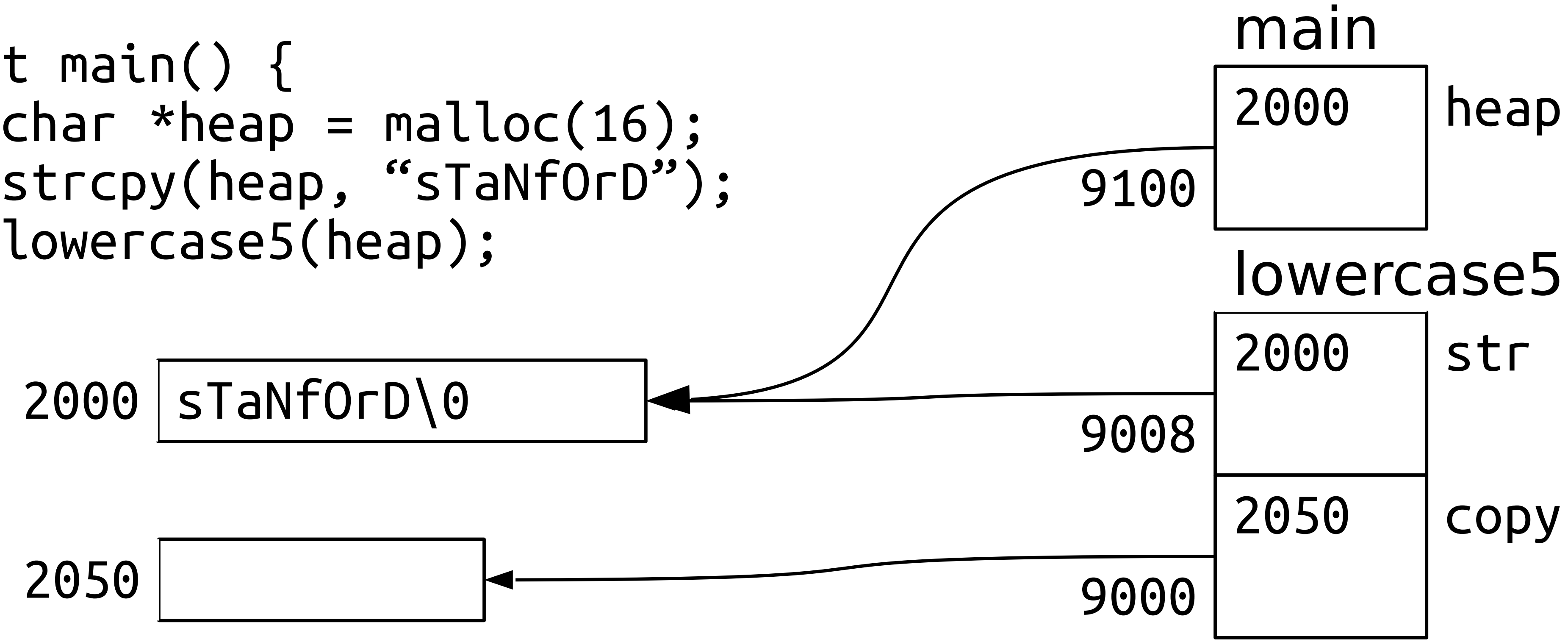
lowercase5

```
void lowercase5(char *str) {
  char *copy = malloc(strlen(str) + 1);
  for (int i = 0; i <= strlen(str); i++)
    copy[i] = tolower(str[i]);
  str = copy;
}

int main() {
  char *heap = malloc(16);
  strcpy(heap, “sTaNfOrD”);
  lowercase5(heap);
}
```



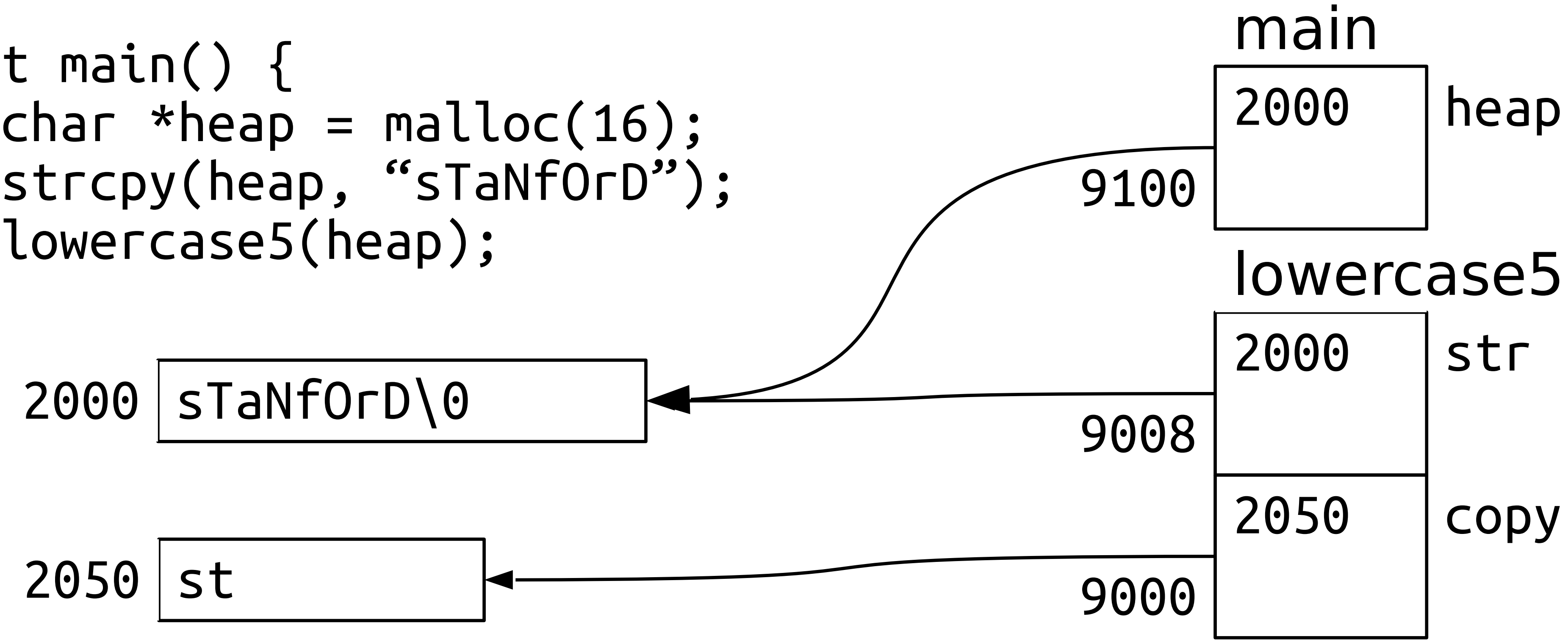
lowercase5

```
void lowercase5(char *str) {
  char *copy = malloc(strlen(str) + 1);
  for (int i = 0; i <= strlen(str); i++)
    copy[i] = tolower(str[i]);
  str = copy;
}

int main() {
  char *heap = malloc(16);
  strcpy(heap, “sTaNfOrD”);
  lowercase5(heap);
}
```



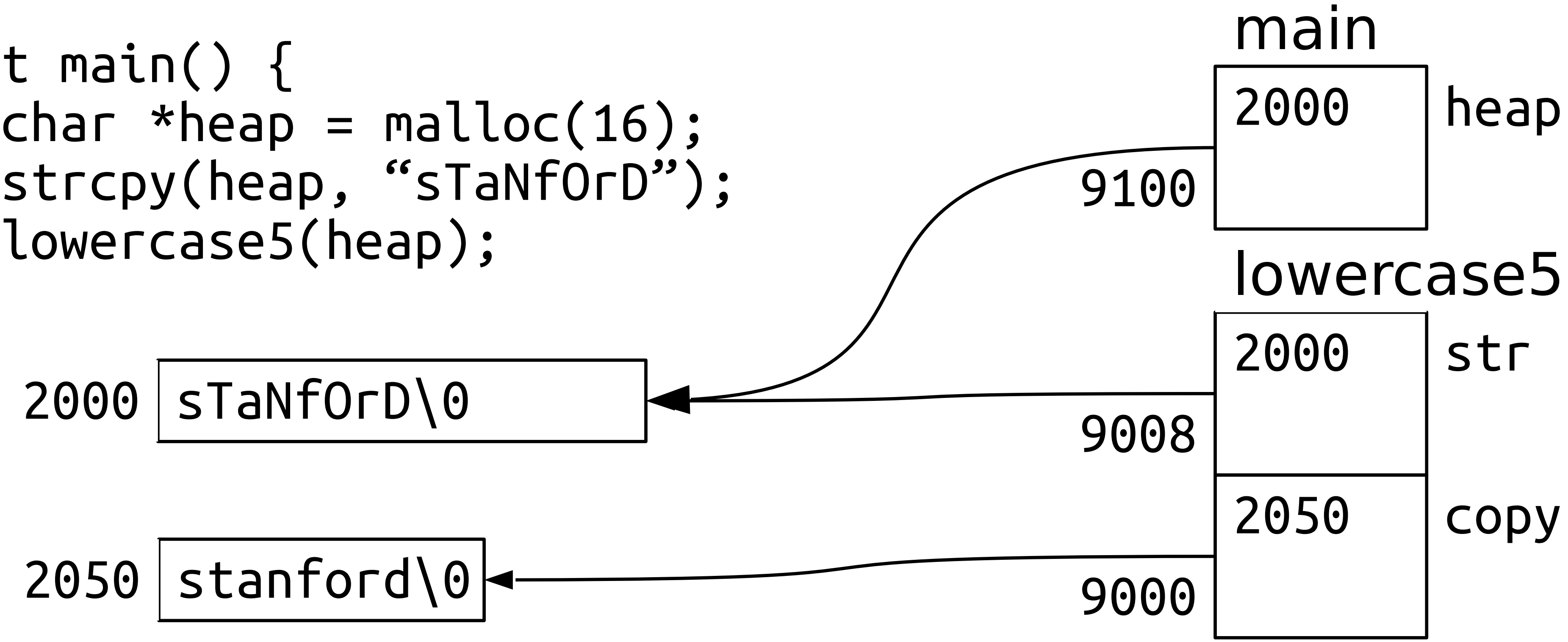
lowercase5

```
void lowercase5(char *str) {
  char *copy = malloc(strlen(str) + 1);
  for (int i = 0; i <= strlen(str); i++)
    copy[i] = tolower(str[i]);
  str = copy;
}

int main() {
  char *heap = malloc(16);
  strcpy(heap, “sTaNfOrD”);
  lowercase5(heap);
}
```



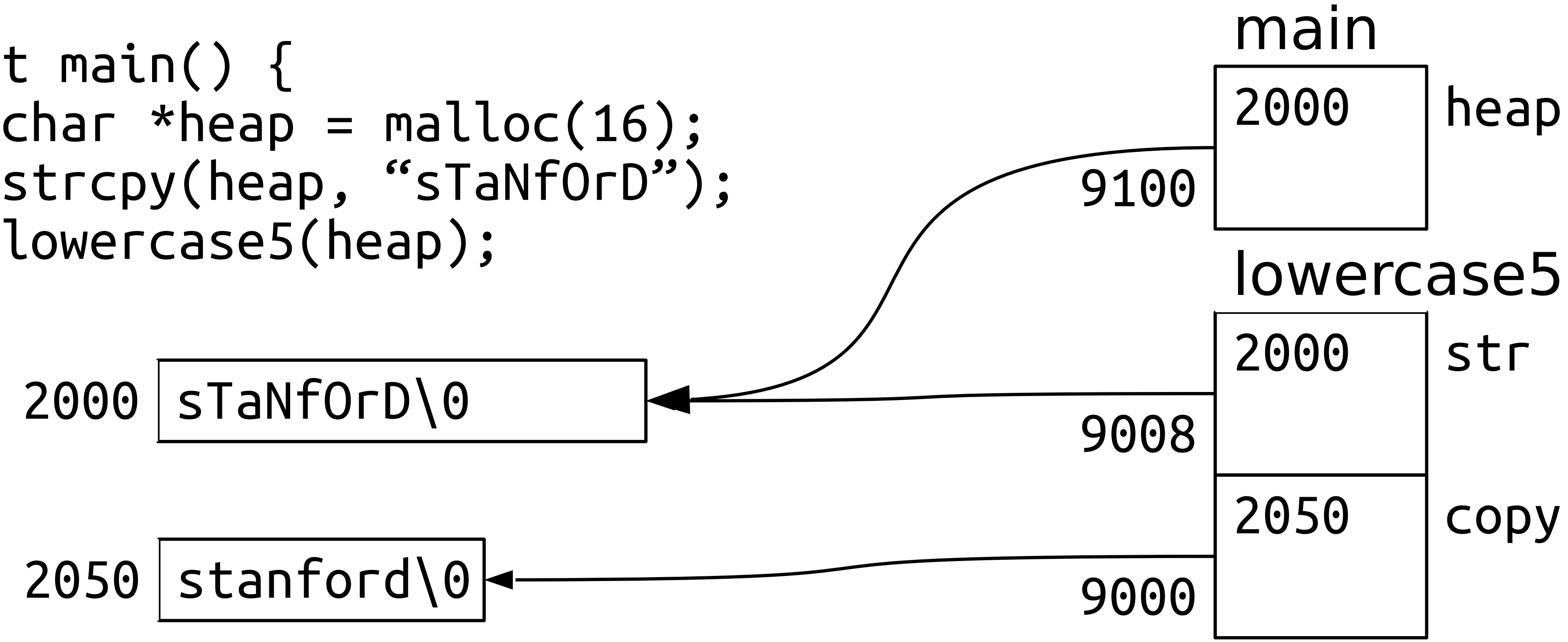
lowercase5

```
void lowercase5(char *str) {
  char *copy = malloc(strlen(str) + 1);
  for (int i = 0; i <= strlen(str); i++)
    copy[i] = tolower(str[i]);
  str = copy;
}

int main() {
  char *heap = malloc(16);
  strcpy(heap, “sTaNfOrD”);
  lowercase5(heap);
}
```



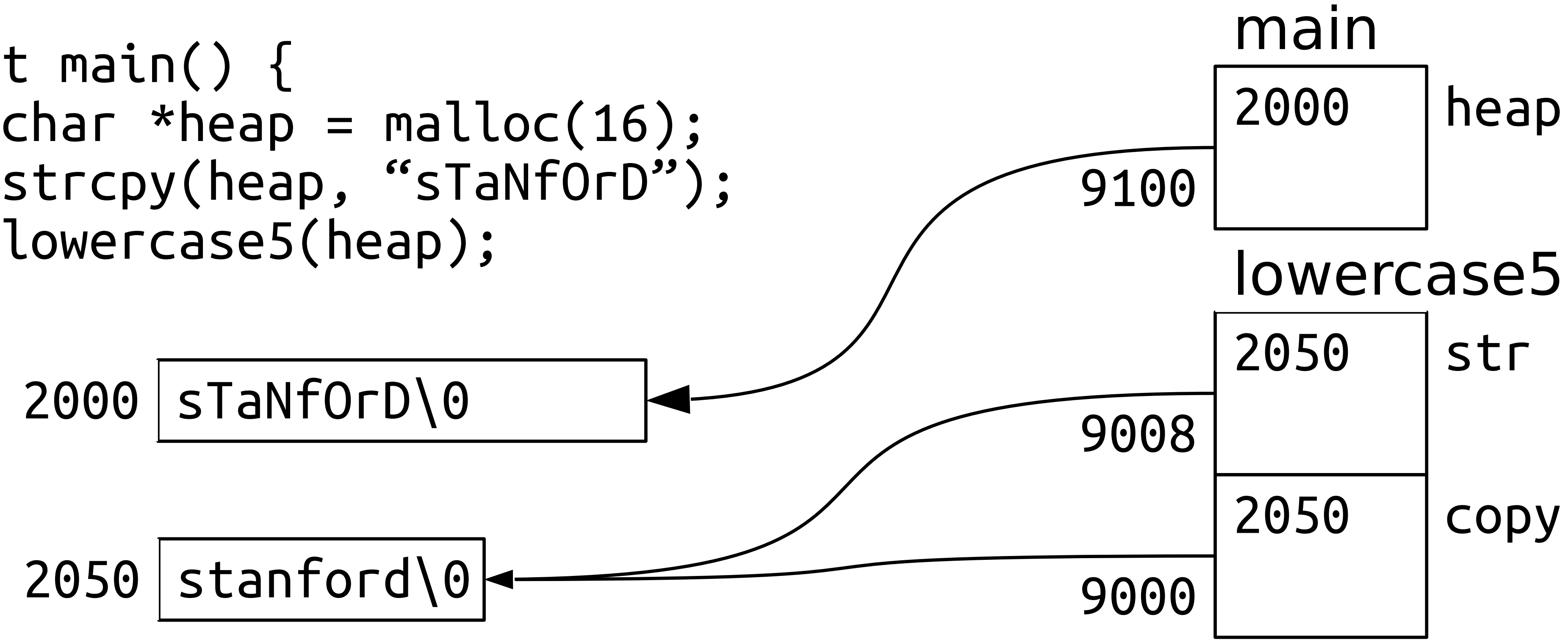
lowercase5

```
void lowercase5(char *str) {
  char *copy = malloc(strlen(str) + 1);
  for (int i = 0; i <= strlen(str); i++)
    copy[i] = tolower(str[i]);
  str = copy;
}

int main() {
  char *heap = malloc(16);
  strcpy(heap, “sTaNfOrD”);
  lowercase5(heap);
}
```



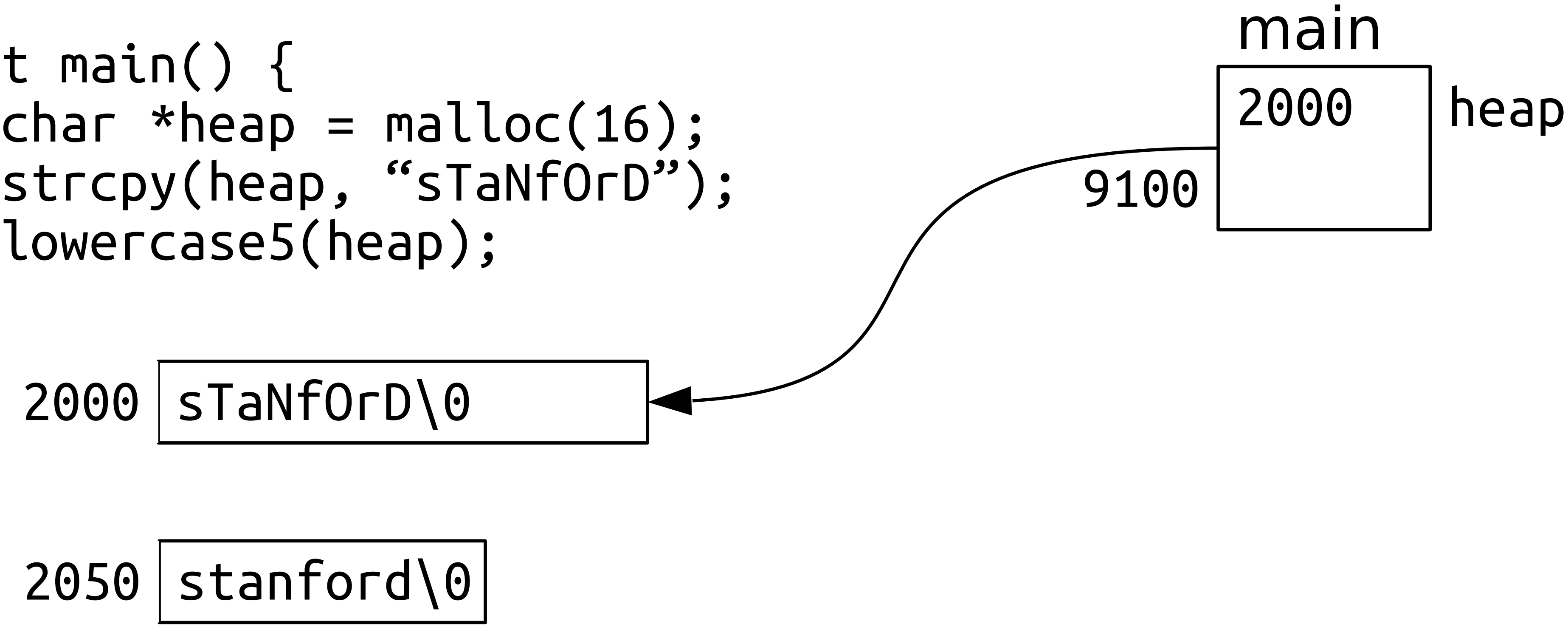
lowercase5

```
void lowercase5(char *str) {
  char *copy = malloc(strlen(str) + 1);
  for (int i = 0; i <= strlen(str); i++)
    copy[i] = tolower(str[i]);
  str = copy;
}

int main() {
  char *heap = malloc(16);
  strcpy(heap, “sTaNfOrD”);
  lowercase5(heap);
}
```



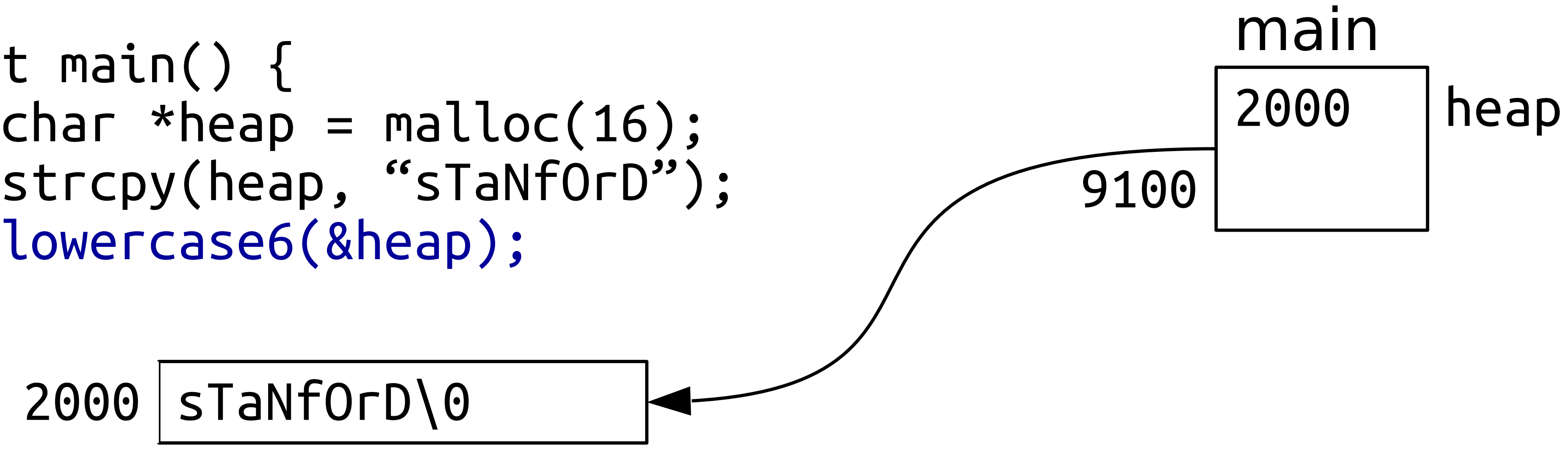
lowercase5

```
void lowercase5(char *str) {
  char *copy = malloc(strlen(str) + 1);
  for (int i = 0; i <= strlen(str); i++)
    copy[i] = tolower(str[i]);
  str = copy;
}

int main() {
  char *heap = malloc(16);
  strcpy(heap, “sTaNfOrD”);
  lowercase5(heap);
}
```



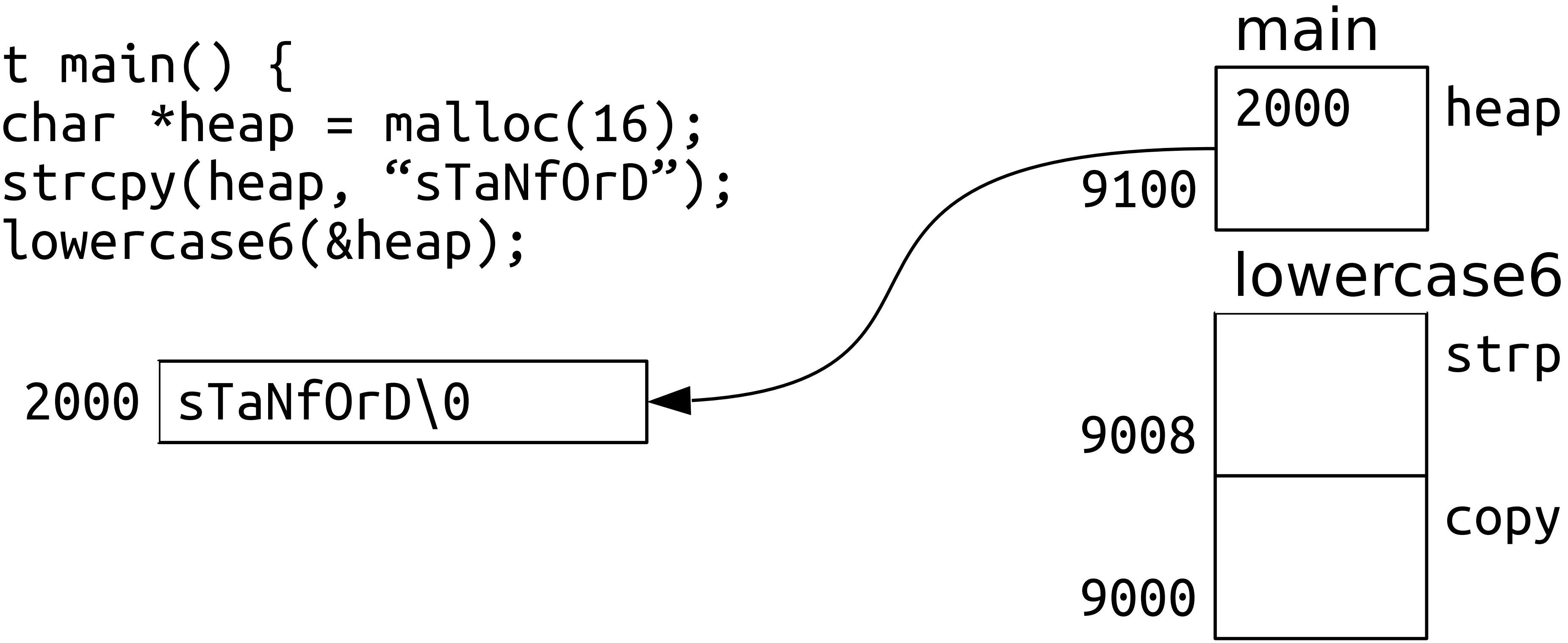
lowercase6

```
void lowercase6(char **strp) {
  char *copy = malloc(strlen(*strp) + 1);
  for (int i = 0; i <= strlen(*strp); i++)
    copy[i] = tolower((*strp)[i]);
  *strp = copy;
}

int main() {
  char *heap = malloc(16);
  strcpy(heap, “sTaNfOrD”);
  lowercase6(&heap);
}
```



lowercase6

```
void lowercase6(char **strp) {
  char *copy = malloc(strlen(*strp) + 1);
  for (int i = 0; i <= strlen(*strp); i++)
    copy[i] = tolower((*strp)[i]);
  *strp = copy;
}

int main() {
  char *heap = malloc(16);
  strcpy(heap, “sTaNfOrD”);
  lowercase6(&heap);
}
```



main

9100	2000	heap

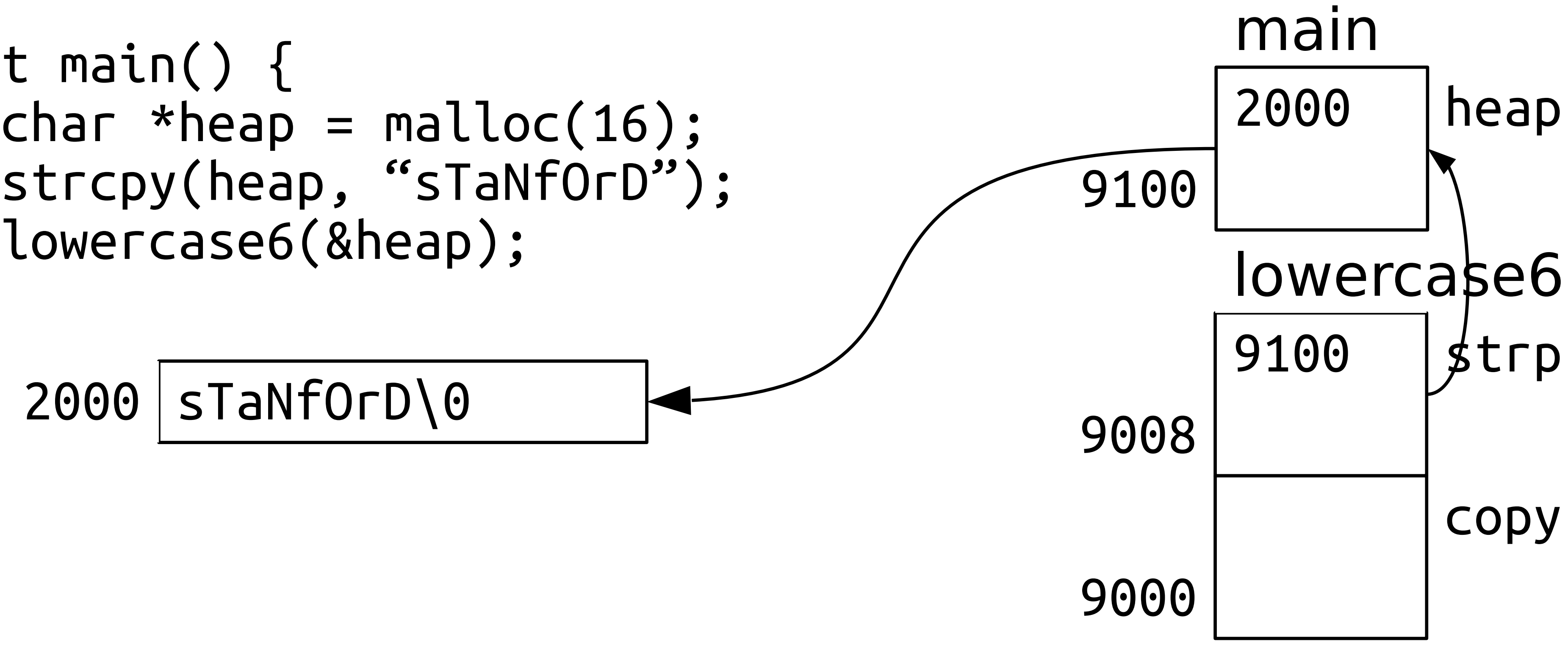
lowercase6

9008		strp
9000		copy

2000 sTaNfOrD\0

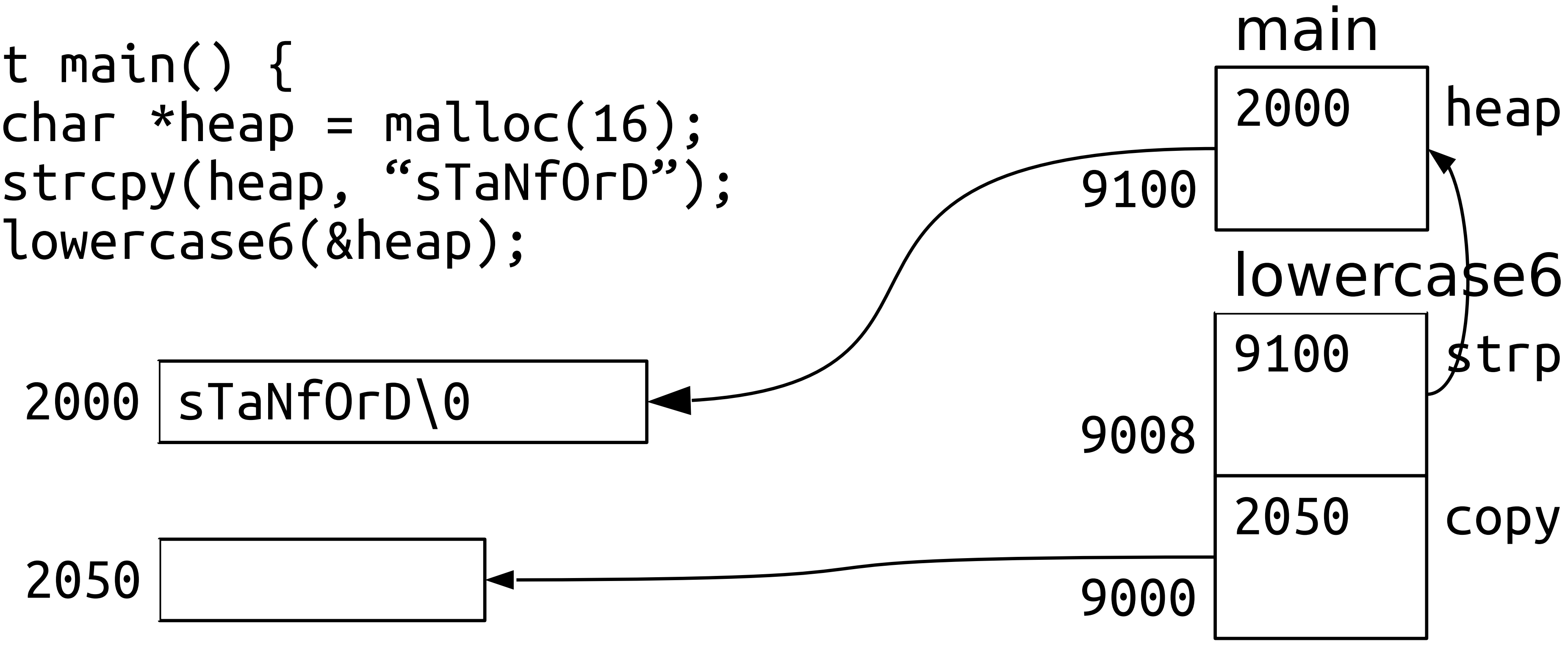
lowercase6

```
void lowercase6(char **strp) {
  char *copy = malloc(strlen(*strp) + 1);
  for (int i = 0; i <= strlen(*strp); i++)
    copy[i] = tolower((*strp)[i]);
  *strp = copy;
}

int main() {
  char *heap = malloc(16);
  strcpy(heap, “sTaNfOrD”);
  lowercase6(&heap);
}
```



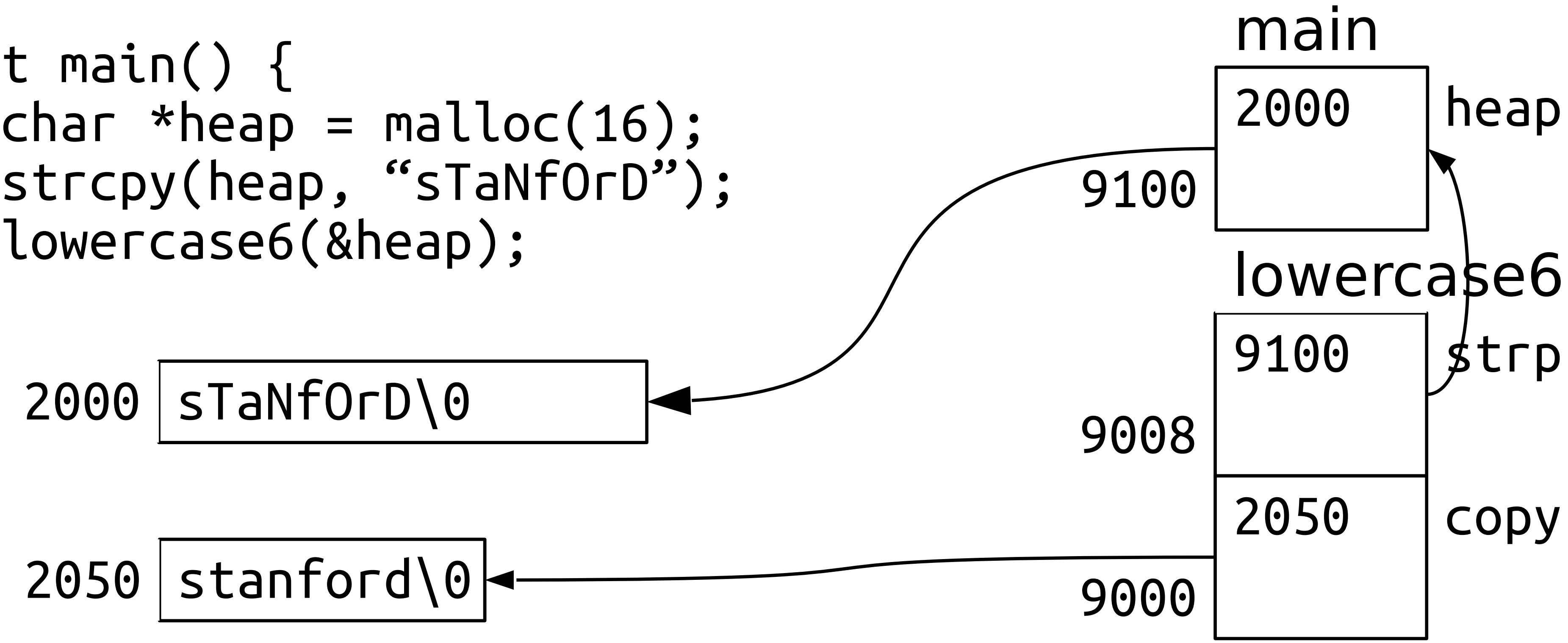
lowercase6

```
void lowercase6(char **strp) {
  char *copy = malloc(strlen(*strp) + 1);
  for (int i = 0; i <= strlen(*strp); i++)
    copy[i] = tolower((*strp)[i]);
  *strp = copy;
}

int main() {
  char *heap = malloc(16);
  strcpy(heap, “sTaNfOrD”);
  lowercase6(&heap);
}
```



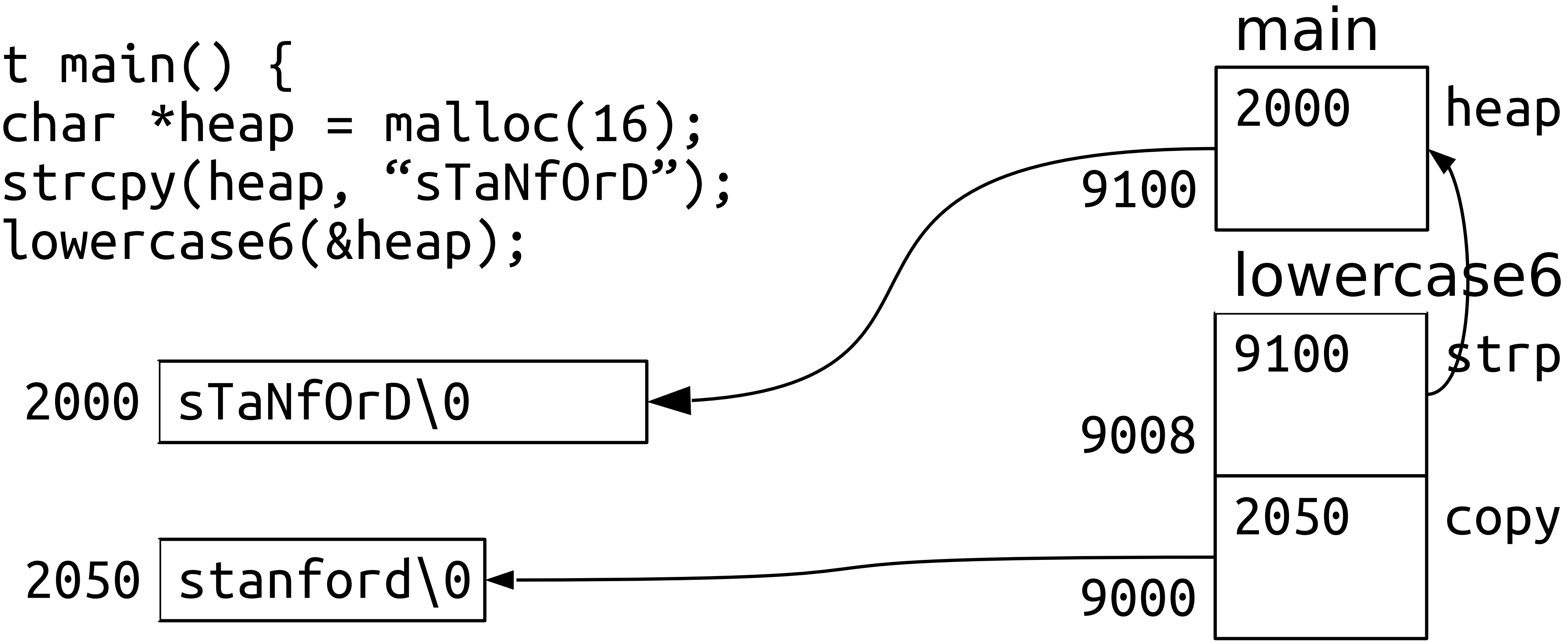
lowercase6

```
void lowercase6(char **strp) {
  char *copy = malloc(strlen(*strp) + 1);
  for (int i = 0; i <= strlen(*strp); i++)
    copy[i] = tolower((*strp)[i]);
  *strp = copy;
}

int main() {
  char *heap = malloc(16);
  strcpy(heap, “sTaNfOrD”);
  lowercase6(&heap);
}
```



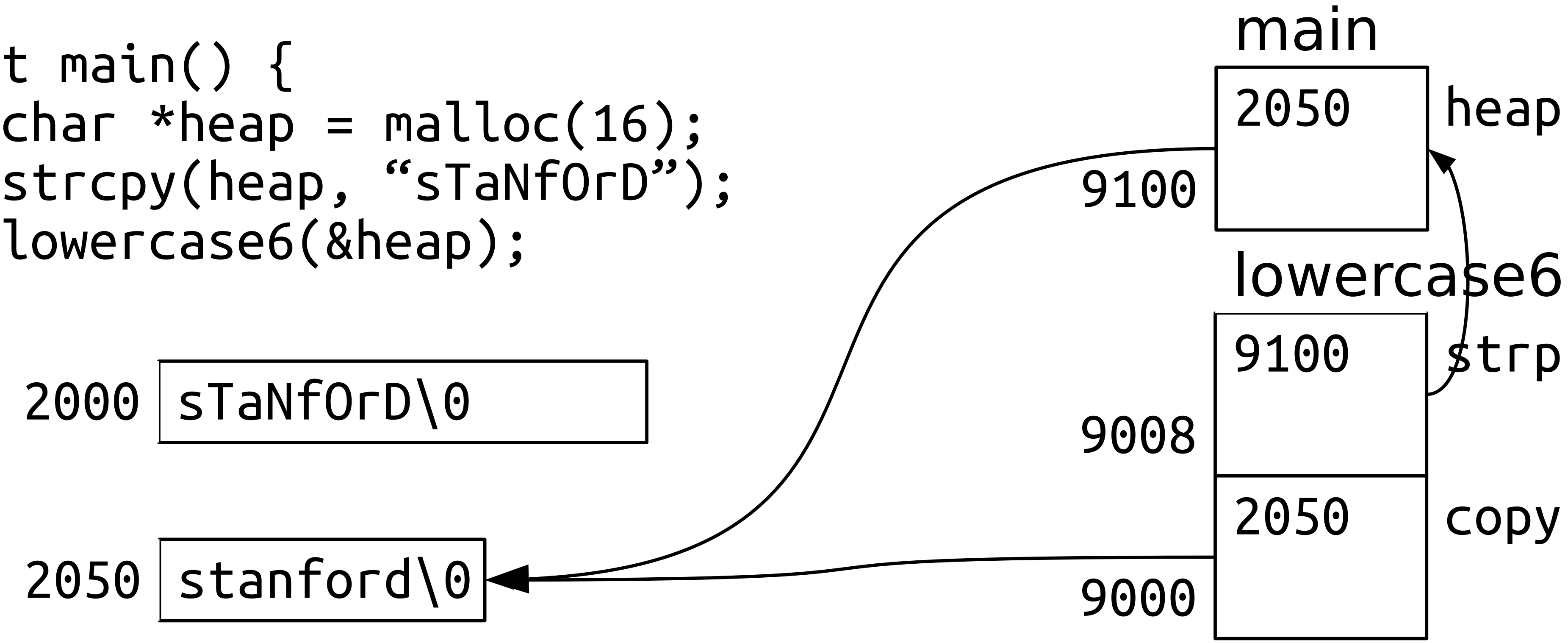
lowercase6

```
void lowercase6(char **strp) {
  char *copy = malloc(strlen(*strp) + 1);
  for (int i = 0; i <= strlen(*strp); i++)
    copy[i] = tolower((*strp)[i]);
  *strp = copy;
}

int main() {
  char *heap = malloc(16);
  strcpy(heap, “sTaNfOrD”);
  lowercase6(&heap);
}
```



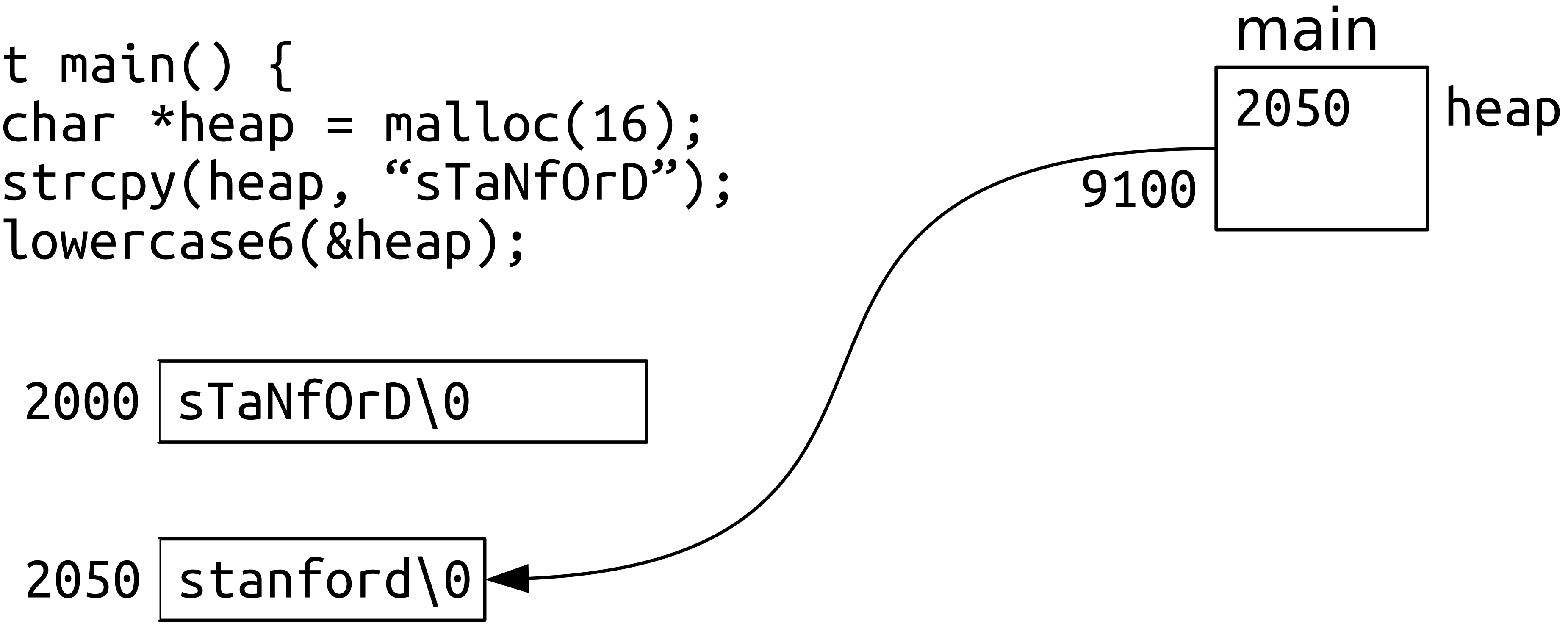
lowercase6

```
void lowercase6(char **strp) {
  char *copy = malloc(strlen(*strp) + 1);
  for (int i = 0; i <= strlen(*strp); i++)
    copy[i] = tolower((*strp)[i]);
  *strp = copy;
}

int main() {
  char *heap = malloc(16);
  strcpy(heap, “sTaNfOrD”);
  lowercase6(&heap);
}
```



lowercase6

```
void lowercase6(char **strp) {
  char *copy = malloc(strlen(*strp) + 1);
  for (int i = 0; i <= strlen(*strp); i++)
    copy[i] = tolower((*strp)[i]);
  *strp = copy;
}

int main() {
  char *heap = malloc(16);
  strcpy(heap, “sTaNfOrD”);
  lowercase6(&heap);
}
```



main

2050

heap

9100

2000 sTaNfOrD\0

2050 stanford\0

Summary

Explore tradeoffs between stack and heap

See pointers used to pass by reference

See and debug common memory errors

- gdb and Valgrind

Next week: generics