
```
1 // Prints two strings' addresses
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 int main(void)
7 {
8     // Get two strings
9     string s = get_string("s: ");
10    string t = get_string("t: ");
11
12    // Print strings' addresses
13    printf("s: %p\n", s);
14    printf("t: %p\n", t);
15 }
```

```
1 // Compares two integers
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 int main(void)
7 {
8     // Get two integers
9     int i = get_int("i: ");
10    int j = get_int("j: ");
11
12    // Compare integers
13    if (i == j)
14    {
15        printf("same\n");
16    }
17    else
18    {
19        printf("different\n");
20    }
21 }
```

```
1 // Compares two strings' addresses
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 int main(void)
7 {
8     // Get two strings
9     string s = get_string("s: ");
10    string t = get_string("t: ");
11
12    // Compare strings' addresses
13    if (s == t)
14    {
15        printf("same\n");
16    }
17    else
18    {
19        printf("different\n");
20    }
21 }
```

```
1 // Compares two strings for equality
2
3 #include <cs50.h>
4 #include <stdio.h>
5 #include <string.h>
6
7 bool compare_strings(string a, string b);
8
9 int main(void)
10 {
11     // Get two strings
12     string s = get_string("s: ");
13     string t = get_string("t: ");
14
15     // Compare strings for equality
16     if (compare_strings(s, t))
17     {
18         printf("same\n");
19     }
20     else
21     {
22         printf("different\n");
23     }
24 }
25
26 bool compare_strings(string a, string b)
27 {
28     // Compare strings' lengths
29     if (strlen(a) != strlen(b))
30     {
31         return false;
32     }
33
34     // Compare strings character by character
35     for (int i = 0, n = strlen(a); i < n; i++)
36     {
37         // Different
38         if (a[i] != b[i])
39         {
40             return false;
41         }
42     }
43
44     // Same
```

```
45     return true;  
46 }
```

```
1 // Compares two strings for equality
2
3 #include <cs50.h>
4 #include <stdio.h>
5 #include <string.h>
6
7 bool compare_strings(char *a, char *b);
8
9 int main(void)
10 {
11     // Get two strings
12     char *s = get_string("s: ");
13     char *t = get_string("t: ");
14
15     // Compare strings for equality
16     if (compare_strings(s, t))
17     {
18         printf("same\n");
19     }
20     else
21     {
22         printf("different\n");
23     }
24 }
25
26 bool compare_strings(char *a, char *b)
27 {
28     // Compare strings' lengths
29     if (strlen(a) != strlen(b))
30     {
31         return false;
32     }
33
34     // Compare strings character by character
35     for (int i = 0, n = strlen(a); i < n; i++)
36     {
37         // Different
38         if (a[i] != b[i])
39         {
40             return false;
41         }
42     }
43
44     // Same
```

```
45     return true;  
46 }
```

```
1 // Compares two strings for equality using strcmp
2
3 #include <cs50.h>
4 #include <stdio.h>
5 #include <string.h>
6
7 int main(void)
8 {
9     // Get two strings
10    char *s = get_string("s: ");
11    char *t = get_string("t: ");
12
13    // Compare strings for equality
14    if (strcmp(s, t) == 0)
15    {
16        printf("same\n");
17    }
18    else
19    {
20        printf("different\n");
21    }
22 }
```



```
1 // Compares two strings for equality while checking for errors
2
3 #include <cs50.h>
4 #include <stdio.h>
5 #include <string.h>
6
7 int main(void)
8 {
9     // Get a string
10    char *s = get_string("s: ");
11    if (s == NULL)
12    {
13        return 1;
14    }
15
16    // Get another string
17    char *t = get_string("t: ");
18    if (t == NULL)
19    {
20        return 1;
21    }
22
23    // Compare strings for equality
24    if (strcmp(s, t) == 0)
25    {
26        printf("same\n");
27    }
28    else
29    {
30        printf("different\n");
31    }
32    return 0;
33 }
```

```
1 // Compares two strings for equality while checking (succinctly) for errors
2
3 #include <cs50.h>
4 #include <stdio.h>
5 #include <string.h>
6
7 int main(void)
8 {
9     // get a string
10    char *s = get_string("s: ");
11    if (!s)
12    {
13        return 1;
14    }
15
16    // get another string
17    char *t = get_string("t: ");
18    if (!t)
19    {
20        return 1;
21    }
22
23    // compare strings for equality
24    if (strcmp(s, t) == 0)
25    {
26        printf("same\n");
27    }
28    else
29    {
30        printf("different\n");
31    }
32    return 0;
33 }
```

```
1 // Capitalizes a string
2
3 #include <cs50.h>
4 #include <ctype.h>
5 #include <stdio.h>
6 #include <string.h>
7
8 int main(void)
9 {
10     // Get a string
11     string s = get_string("s: ");
12
13     // Copy string's address
14     string t = s;
15
16     // Capitalize first letter in string
17     if (strlen(t) > 0)
18     {
19         t[0] = toupper(t[0]);
20     }
21
22     // Print string twice
23     printf("s: %s\n", s);
24     printf("t: %s\n", t);
25 }
```

```
1 // Capitalizes a copy of a string while checking for errors
2
3 #include <cs50.h>
4 #include <ctype.h>
5 #include <stdio.h>
6 #include <string.h>
7
8 int main(void)
9 {
10     // Get a string
11     char *s = get_string("s: ");
12     if (!s)
13     {
14         return 1;
15     }
16
17     // Allocate memory for another string
18     char *t = malloc((strlen(s) + 1) * sizeof(char));
19     if (!t)
20     {
21         return 1;
22     }
23
24     // Copy string into memory
25     for (int i = 0, n = strlen(s); i <= n; i++)
26     {
27         t[i] = s[i];
28     }
29
30     // Capitalize first letter in copy
31     if (strlen(t) > 0)
32     {
33         t[0] = toupper(t[0]);
34     }
35
36     // Print strings
37     printf("s: %s\n", s);
38     printf("t: %s\n", t);
39
40     // Free memory
41     free(t);
42     return 0;
43 }
```

```
1 // Capitalizes a copy of a string using strcpy while checking for errors
2
3 #include <cs50.h>
4 #include <ctype.h>
5 #include <stdio.h>
6 #include <string.h>
7
8 int main(void)
9 {
10     // Get a string
11     char *s = get_string("s: ");
12     if (!s)
13     {
14         return 1;
15     }
16
17     // Allocate memory for another string
18     char *t = malloc((strlen(s) + 1) * sizeof(char));
19     if (!t)
20     {
21         return 1;
22     }
23
24     // Copy string into memory
25     strcpy(t, s);
26
27     // Capitalize first letter in copy
28     if (strlen(t) > 0)
29     {
30         t[0] = toupper(t[0]);
31     }
32
33     // Print strings
34     printf("s: %s\n", s);
35     printf("t: %s\n", t);
36
37     // Free memory
38     free(t);
39     return 0;
40 }
```

```
1  #include <cs50.h>
2  #include <stdio.h>
3
4  int main(void)
5  {
6      string name = get_string("Name: ");
7      printf("hello, %s\n", name);
8  }
```

```
1 // http://valgrind.org/docs/manual/quick-start.html#quick-start.prepare
2
3 #include <stdlib.h>
4
5 void f(void)
6 {
7     int *x = malloc(10 * sizeof(int));
8     x[10] = 0;
9 }
10
11 int main(void)
12 {
13     f();
14     return 0;
15 }
```

```
1 // Fails to swap two integers
2
3 #include <stdio.h>
4
5 void swap(int a, int b);
6
7 int main(void)
8 {
9     int x = 1;
10    int y = 2;
11
12    printf("x is %i, y is %i\n", x, y);
13    swap(x, y);
14    printf("x is %i, y is %i\n", x, y);
15 }
16
17 void swap(int a, int b)
18 {
19     int tmp = a;
20     a = b;
21     b = tmp;
22 }
```

```
1 // Gets an int from user using scanf
2
3 #include <stdio.h>
4
5 int main(void)
6 {
7     int x;
8     printf("x: ");
9     scanf("%i", &x);
10    printf("x: %i\n", x);
11 }
```

```
1 // Incorrectly gets a string from user using scanf
2
3 #include <stdio.h>
4
5 int main(void)
6 {
7     char *s;
8     printf("s: ");
9     scanf("%s", s);
10    printf("s: %s\n", s);
11 }
```

```
1 // Dangerously gets a string from user using scanf
2
3 #include <stdio.h>
4
5 int main(void)
6 {
7     char s[5];
8     printf("s: ");
9     scanf("%s", s);
10    printf("s: %s\n", s);
11 }
```

```
1 // Prints a string's characters using square brackets
2
3 #include <cs50.h>
4 #include <stdio.h>
5 #include <string.h>
6
7 int main(void)
8 {
9     // Get a string
10    char *s = get_string("string: ");
11    if (!s)
12    {
13        return 1;
14    }
15
16    // Print string, one character per line
17    for (int i = 0, n = strlen(s); i < n; i++)
18    {
19        printf("%c\n", s[i]);
20    }
21    return 0;
22 }
```

```
1 // Prints a string's characters using pointer arithmetic
2
3 #include <cs50.h>
4 #include <stdio.h>
5 #include <string.h>
6
7 int main(void)
8 {
9     // Get a string
10    char *s = get_string("string: ");
11    if (!s)
12    {
13        return 1;
14    }
15
16    // Print string, one character per line
17    for (int i = 0, n = strlen(s); i < n; i++)
18    {
19        printf("%c\n", *(s + i));
20    }
21    return 0;
22 }
```

```
1 // Demonstrates lack of structs
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 int main(void)
7 {
8     // Space for students
9     int enrollment = get_int("Enrollment: ");
10    string names[enrollment];
11    string dorms[enrollment];
12
13    // Prompt for students' names and dorms
14    for (int i = 0; i < enrollment; i++)
15    {
16        names[i] = get_string("Name: ");
17        dorms[i] = get_string("Dorm: ");
18    }
19
20    // Print students' names and dorms
21    for (int i = 0; i < enrollment; i++)
22    {
23        printf("%s is in %s.\n", names[i], dorms[i]);
24    }
25 }
```

```
1 // Demonstrates structs
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 #include "struct.h"
7
8 int main(void)
9 {
10     // Space for students
11     int enrollment = get_int("Enrollment: ");
12     student students[enrollment];
13
14     // Prompt for students' names and dorms
15     for (int i = 0; i < enrollment; i++)
16     {
17         students[i].name = get_string("Name: ");
18         students[i].dorm = get_string("Dorm: ");
19     }
20
21     // Print students' names and dorms
22     for (int i = 0; i < enrollment; i++)
23     {
24         printf("%s is in %s.\n", students[i].name, students[i].dorm);
25     }
26 }
```

```
1 // Demonstrates file I/O
2
3 #include <cs50.h>
4 #include <stdio.h>
5 #include <stdlib.h>
6
7 #include "struct.h"
8
9 int main(void)
10 {
11     // Space for students
12     int enrollment = get_int("Enrollment: ");
13     student students[enrollment];
14
15     // Prompt for students' names and dorms
16     for (int i = 0; i < enrollment; i++)
17     {
18         students[i].name = get_string("Name: ");
19         students[i].dorm = get_string("Dorm: ");
20     }
21
22     // Save students to disk
23     FILE *file = fopen("students.csv", "w");
24     if (file)
25     {
26         for (int i = 0; i < enrollment; i++)
27         {
28             fprintf(file, "%s,%s\n", students[i].name, students[i].dorm);
29         }
30         fclose(file);
31     }
32 }
```

```
1 // Represents a student
2
3 typedef struct
4 {
5     char *name;
6     char *dorm;
7 }
8 student;
```

```
1 // Swaps two integers using pointers
2
3 #include <stdio.h>
4
5 void swap(int *a, int *b);
6
7 int main(void)
8 {
9     int x = 1;
10    int y = 2;
11
12    printf("x is %i, y is %i\n", x, y);
13    swap(&x, &y);
14    printf("x is %i, y is %i\n", x, y);
15 }
16
17 void swap(int *a, int *b)
18 {
19     int tmp = *a;
20     *a = *b;
21     *b = tmp;
22 }
```