
```
1 // Prints an integer
2
3 #include <stdio.h>
4
5 int main(void)
6 {
7     int n = 50;
8     printf("%i\n", n);
9 }
```

```
1 // Prints an integer's address
2
3 #include <stdio.h>
4
5 int main(void)
6 {
7     int n = 50;
8     printf("%p\n", &n);
9 }
```

```
1 // Prints an integer via its address
2
3 #include <stdio.h>
4
5 int main(void)
6 {
7     int n = 50;
8     printf("%i\n", *&n);
9 }
```

```
1 // Stores and prints an integer's address
2
3 #include <stdio.h>
4
5 int main(void)
6 {
7     int n = 50;
8     int *p = &n;
9     printf("%p\n", p);
10 }
```

```
1 // Stores and prints an integer via its address
2
3 #include <stdio.h>
4
5 int main(void)
6 {
7     int n = 50;
8     int *p = &n;
9     printf("%i\n", *p);
10 }
```

```
1 // Prints a string
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 int main(void)
7 {
8     string s = "HI!";
9     printf("%s\n", s);
10 }
```

```
1 // Prints a string's address
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 int main(void)
7 {
8     string s = "HI!";
9     printf("%p\n", s);
10 }
```

```
1 // Prints a string's address as well the addresses of its chars
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 int main(void)
7 {
8     string s = "HI!";
9     printf("%p\n", s);
10    printf("%p\n", &s[0]);
11    printf("%p\n", &s[1]);
12    printf("%p\n", &s[2]);
13    printf("%p\n", &s[3]);
14 }
```

```
1 // Prints a string's chars
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 int main(void)
7 {
8     string s = "HI!";
9     printf("%c\n", s[0]);
10    printf("%c\n", s[1]);
11    printf("%c\n", s[2]);
12 }
```

```
1 // Stores and prints a string without using the CS50 Library
2
3 #include <stdio.h>
4
5 int main(void)
6 {
7     char *s = "HI!";
8     printf("%c\n", s[0]);
9     printf("%c\n", s[1]);
10    printf("%c\n", s[2]);
11 }
```

```
1 // Stores and prints a string's address via pointer arithmetic
2
3 #include <stdio.h>
4
5 int main(void)
6 {
7     char *s = "HI!";
8     printf("%c\n", *s);
9     printf("%c\n", *(s+1));
10    printf("%c\n", *(s+2));
11 }
```

```
1 // Prints an array using pointer arithmetic
2
3 #include <stdio.h>
4
5 int main(void)
6 {
7     // An array of numbers
8     int numbers[] = {4, 6, 8, 2, 7, 5, 0};
9
10    // Print numbers
11    printf("%i\n", *numbers);
12    printf("%i\n", *(numbers+1));
13    printf("%i\n", *(numbers+2));
14    printf("%i\n", *(numbers+3));
15    printf("%i\n", *(numbers+4));
16    printf("%i\n", *(numbers+5));
17    printf("%i\n", *(numbers+6));
18 }
```

```
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 using strcmp
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
13    if (strcmp(s, t) == 0)
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     char *s = get_string("s: ");
10    char *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 // Prints two strings
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 int main(void)
7 {
8     // Get two strings
9     char *s = get_string("s: ");
10    char *t = get_string("t: ");
11
12    // Print strings
13    printf("%s\n", s);
14    printf("%s\n", t);
15 }
```



```
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     char *s = get_string("s: ");
10    char *t = get_string("t: ");
11
12    // Print strings' addresses
13    printf("%p\n", s);
14    printf("%p\n", t);
15 }
```

```
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
2
3 #include <cs50.h>
4 #include <ctype.h>
5 #include <stdio.h>
6 #include <stdlib.h>
7 #include <string.h>
8
9 int main(void)
10 {
11     // Get a string
12     char *s = get_string("s: ");
13
14     // Allocate memory for another string
15     char *t = malloc(strlen(s) + 1);
16
17     // Copy string into memory
18     for (int i = 0, n = strlen(s); i <= n; i++)
19     {
20         t[i] = s[i];
21     }
22
23     // Capitalize copy
24     t[0] = toupper(t[0]);
25
26     // Print strings
27     printf("s: %s\n", s);
28     printf("t: %s\n", t);
29 }
```

```
1 // Capitalizes a copy of a string using strcpy
2
3 #include <cs50.h>
4 #include <ctype.h>
5 #include <stdio.h>
6 #include <stdlib.h>
7 #include <string.h>
8
9 int main(void)
10 {
11     // Get a string
12     char *s = get_string("s: ");
13
14     // Allocate memory for another string
15     char *t = malloc(strlen(s) + 1);
16
17     // Copy string into memory
18     strcpy(t, s);
19
20     // Capitalize copy
21     t[0] = toupper(t[0]);
22
23     // Print strings
24     printf("s: %s\n", s);
25     printf("t: %s\n", t);
26 }
```

```
1 // Capitalizes a copy of a string without memory errors
2
3 #include <cs50.h>
4 #include <ctype.h>
5 #include <stdio.h>
6 #include <stdlib.h>
7 #include <string.h>
8
9 int main(void)
10 {
11     // Get a string
12     char *s = get_string("s: ");
13     if (s == NULL)
14     {
15         return 1;
16     }
17
18     // Allocate memory for another string
19     char *t = malloc(strlen(s) + 1);
20     if (t == NULL)
21     {
22         return 1;
23     }
24
25     // Copy string into memory
26     strcpy(t, s);
27
28     // Capitalize copy
29     if (strlen(t) > 0)
30     {
31         t[0] = toupper(t[0]);
32     }
33
34     // Print strings
35     printf("s: %s\n", s);
36     printf("t: %s\n", t);
37
38     // Free memory
39     free(t);
40     return 0;
41 }
```

```
1 // Demonstrates memory errors via valgrind
2
3 #include <stdio.h>
4 #include <stdlib.h>
5
6 int main(void)
7 {
8     int *x = malloc(3 * sizeof(int));
9     x[1] = 72;
10    x[2] = 73;
11    x[3] = 33;
12 }
```

```
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 // 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 }
```

```
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[4];
8     printf("s: ");
9     scanf("%s", s);
10    printf("s: %s\n", s);
11 }
```

```
1 // Saves names and numbers to a CSV file
2
3 #include <cs50.h>
4 #include <stdio.h>
5 #include <string.h>
6
7 int main(void)
8 {
9     // Open CSV file
10    FILE *file = fopen("phonebook.csv", "a");
11    if (!file)
12    {
13        return 1;
14    }
15
16    // Get name and number
17    string name = get_string("Name: ");
18    string number = get_string("Number: ");
19
20    // Print to file
21    fprintf(file, "%s,%s\n", name, number);
22
23    // Close file
24    fclose(file);
25 }
```

```
1 // Detects if a file is a JPEG
2
3 #include <stdint.h>
4 #include <stdio.h>
5
6 typedef uint8_t BYTE;
7
8 int main(int argc, char *argv[])
9 {
10     // Check usage
11     if (argc != 2)
12     {
13         return 1;
14     }
15
16     // Open file
17     FILE *file = fopen(argv[1], "r");
18     if (!file)
19     {
20         return 1;
21     }
22
23     // Read first three bytes
24     BYTE bytes[3];
25     fread(bytes, sizeof(BYTE), 3, file);
26
27     // Check first three bytes
28     if (bytes[0] == 0xff && bytes[1] == 0xd8 && bytes[2] == 0xff)
29     {
30         printf("Maybe\n");
31     }
32     else
33     {
34         printf("No\n");
35     }
36
37     // Close file
38     fclose(file);
39 }
```

```
1 // Copies a file
2
3 #include <stdint.h>
4 #include <stdio.h>
5 #include <stdlib.h>
6
7 typedef uint8_t BYTE;
8
9 int main(int argc, char *argv[])
10 {
11     // Ensure proper usage
12     if (argc != 3)
13     {
14         fprintf(stderr, "Usage: copy SOURCE DESTINATION\n");
15         return 1;
16     }
17
18     // open input file
19     FILE *source = fopen(argv[1], "r");
20     if (source == NULL)
21     {
22         printf("Could not open %s.\n", argv[1]);
23         return 1;
24     }
25
26     // Open output file
27     FILE *destination = fopen(argv[2], "w");
28     if (destination == NULL)
29     {
30         fclose(source);
31         printf("Could not create %s.\n", argv[2]);
32         return 1;
33     }
34
35     // Copy source to destination, one BYTE at a time
36     BYTE buffer;
37     while (fread(&buffer, sizeof(BYTE), 1, source))
38     {
39         fwrite(&buffer, sizeof(BYTE), 1, destination);
40     }
41
42     // Close files
43     fclose(source);
44     fclose(destination);
```

```
45     return 0;  
46 }
```