```c
// Compara dois inteiros

#include <cs50.h>
#include <stdio.h>

int main(void)
{
    // Recebe dois inteiros
    int i = get_int("i: ");
    int j = get_int("j: ");

    // Compara inteiros
    if (i == j)
    {
        printf("Iguais\n");
    }
    else
    {
        printf("Diferentes\n");
    }
}
```

```c
// Compara os endereços de duas strings

#include <cs50.h>
#include <stdio.h>

int main(void)
{
    // Recebe duas strings
    string s = get_string("s: ");
    string t = get_string("t: ");

    // Compara os endereços das strings
    if (s == t)
    {
        printf("Iguais\n");
    }
    else
    {
        printf("Diferentes\n");
    }
}
```

```c
// Imprime duas strings

#include <cs50.h>
#include <stdio.h>

int main(void)
{
    // Recebe duas strings
    string s = get_string("s: ");
    string t = get_string("t: ");

    // Imprime strings
    printf("%s\n", s);
    printf("%s\n", t);
}
```

```c
// Imprime os endereços de duas strings

#include <cs50.h>
#include <stdio.h>

int main(void)
{
    // Recebe duas strings
    string s = get_string("s: ");
    string t = get_string("t: ");

    // Imprime os endereços das strings
    printf("%p\n", s);
    printf("%p\n", t);
}
```

```c
// Transforma a primeira letra de uma string em maiúscula

#include <cs50.h>
#include <ctype.h>
#include <stdio.h>
#include <string.h>

int main(void)
{
    // Recebe uma string
    string s = get_string("s: ");

    // Copia o endereço da string
    string t = s;

    // Transforma a primeira letra da string em maiúscula
    if (strlen(t) > 0)
    {
        t[0] = toupper(t[0]);
    }

    // Imprime a string duas vezes
    printf("s: %s\n", s);
    printf("t: %s\n", t);
}
```

```c
// Transforma a primeira letra da cópia de uma string em maiúscula

#include <cs50.h>
#include <ctype.h>
#include <stdio.h>
#include <string.h>

int main(void)
{
    // Recebe a string
    char *s = get_string("s: ");

    // Aloca memória para outra string
    char *t = malloc(strlen(s) + 1);

    // Cria uma cópia da string 's' na memória alocada
    for (int i = 0, n = strlen(s); i <= n; i++)
    {
        t[i] = s[i];
    }

    // Transforma a primeira letra da cópia em maiúscula
    t[0] = toupper(t[0]);

    // Imprime as strings
    printf("s: %s\n", s);
    printf("t: %s\n", t);
}
```

```c
// Transforma a primeira letra da cópia de uma string em maiúscula usando strcpy

#include <cs50.h>
#include <ctype.h>
#include <stdio.h>
#include <string.h>

int main(void)
{
    // Recebe uma string
    char *s = get_string("s: ");

    // Aloca memória para outra string
    char *t = malloc(strlen(s) + 1);

    // Cria uma cópia da string 's' na memória alocada
    strcpy(t, s);

    // Transforma a primeira letra da cópia em maiúscula
    t[0] = toupper(t[0]);

    // Imprime as strings
    printf("s: %s\n", s);
    printf("t: %s\n", t);
}
```

```c
// Transforma a primeira letra da cópia de uma string em maiúscula sem causar erros de memória

#include <cs50.h>
#include <ctype.h>
#include <stdio.h>
#include <string.h>

int main(void)
{
    // Recebe uma string
    char *s = get_string("s: ");
    if (s != NULL)
    {
        return 1;
    }

    // Aloca memória para outra string
    char *t = malloc(strlen(s) + 1);
    if (t != NULL)
    {
        return 1;
    }

    // Cria uma cópia da string 's' na memória alocada
    strcpy(t, s);

    // Transforma a primeira letra da cópia da string em maiúscula
    t[0] = toupper(t[0]);

    // Imprime as strings
    printf("s: %s\n", s);
    printf("t: %s\n", t);

    // Libera a memória
    free(t);
    return 0;
}
```

```c
// Imprime um inteiro

#include <stdio.h>

int main(void)
{
    int n = 50;
    printf("%i\n", n);
}
```

```c
// Imprime o endereço de um inteiro

#include <stdio.h>

int main(void)
{
    int n = 50;
    printf("%p\n", &n);
}
```

```c
// Imprime um inteiro através de seu endereço

#include <stdio.h>

int main(void)
{
    int n = 50;
    printf("%i\n", *&n);
}
```

```c
// Armazena e imprime o endereço de um inteiro

#include <stdio.h>

int main(void)
{
    int n = 50;
    int *p = &n;
    printf("%p\n", p);
}
```

```c
// Armazena e imprime um inteiro através de seu endereço

#include <stdio.h>

int main(void)
{
    int n = 50;
    int *p = &n;
    printf("%i\n", *p);
}
```

```c
// Imprime uma string

#include <cs50.h>
#include <stdio.h>

int main(void)
{
    string s = "SARA";
    printf("%s\n", s);
}
```

```c
// Imprime o endereço de uma string

#include <cs50.h>
#include <stdio.h>

int main(void)
{
    string s = "SARA";
    printf("%p\n", s);
}
```

```c
// Imprime o endereço de uma string assim como os endereços de seus caracteres

#include <cs50.h>
#include <stdio.h>

int main(void)
{
    string s = "SARA";
    printf("%p\n", s);
    printf("%p\n", &s[0]);
    printf("%p\n", &s[1]);
    printf("%p\n", &s[2]);
    printf("%p\n", &s[3]);
    printf("%p\n", &s[4]);
}
```

```c
// Imprime os caracteres de uma string

#include <cs50.h>
#include <stdio.h>

int main(void)
{
    string s = "SARA";
    printf("%c\n", s[0]);
    printf("%c\n", s[1]);
    printf("%c\n", s[2]);
    printf("%c\n", s[3]);
}
```

```c
// Armazena e imprime uma string sem usar a biblioteca do CS50

#include <stdio.h>

int main(void)
{
    char *s = "SARA";
    printf("%c\n", s[0]);
    printf("%c\n", s[1]);
    printf("%c\n", s[2]);
    printf("%c\n", s[3]);
}
```

```c
// Armazena e imprime o endereço de uma string usando aritmética em ponteiros

#include <stdio.h>

int main(void)
{
    char *s = "SARA";
    printf("%c\n", *s);
    printf("%c\n", *(s+1));
    printf("%c\n", *(s+2));
    printf("%c\n", *(s+3));
}
```

```c
// Detecta se um arquivo é JPEG

#include <stdio.h>

int main(int argc, char *argv[])
{
    // Checa uso correto do programa
    if (argc != 2)
    {
        return 1;
    }

    // Abre arquivo em modo leitura
    FILE *arquivo = fopen(argv[1], "r");
    if (!arquivo)
    {
        return 1;
    }

    // Lê os primeiros três bytes
    unsigned char bytes[3];
    fread(bytes, 3, 1, arquivo);

    // Checa os primeiros três bytes
    if (bytes[0] == 0xff && bytes[1] == 0xd8 && bytes[2] == 0xff)
    {
        printf("Talvez\n");
    }
    else
    {
        printf("Não\n");
    }

    // Fecha o arquivo
    fclose(arquivo);
}
```

```c
// Salva nomes e números para um arquivo CSV

#include <cs50.h>
#include <stdio.h>
#include <string.h>

int main(void)
{
    // Abre arquivo CSV
    FILE *arquivo = fopen("listatelefonica.csv", "a");
    if (!arquivo)
    {
        return 1;
    }

    // Recebe nome e número
    string nome = get_string("Nome: ");
    string numero = get_string("Número: ");

    // Imprime nome e número no arquivo
    fprintf(arquivo, "%s,%s\n", nome, numero);

    // Fecha arquivo
    fclose(arquivo);
}
```

1

```c
// Leitura sobre a ferramenta Valgrind:
// https://www.ic.unicamp.br/~rafael/cursos/2s2017/mc202/valgrind.html

#include <stdlib.h>

void f(void)
{
    int *x = malloc(10 * sizeof(int));
    x[10] = 0;
}

int main(void)
{
    f();
    return 0;
}
```

```c
// Falha ao tentar trocar dois inteiros

#include <stdio.h>

void trocar(int a, int b);

int main(void)
{
    int x = 1;
    int y = 2;

    printf("x é %i, y é %i\n", x, y);
    trocar(x, y);
    printf("x é %i, y é %i\n", x, y);
}

void trocar(int a, int b)
{
    int tmp = a;
    a = b;
    b = tmp;
}
```

```c
// Recebe um inteiro do usuário usando scanf

#include <stdio.h>

int main(void)
{
    int x;
    printf("x: ");
    scanf("%i", &x);
    printf("x: %i\n", x);
}
```

```c
// Recebe uma string do usuário usando scanf de forma INCORRETA

#include <stdio.h>

int main(void)
{
    char *s;
    printf("s: ");
    scanf("%s", s);
    printf("s: %s\n", s);
}
```

```c
// Recebe uma string do usuário usando scanf de forma PERIGOSA

#include <stdio.h>

int main(void)
{
    char s[5];
    printf("s: ");
    scanf("%s", s);
    printf("s: %s\n", s);
}
```

```c
// Troca dois inteiros usando ponteiros

#include <stdio.h>

void trocar(int *a, int *b);

int main(void)
{
    int x = 1;
    int y = 2;

    printf("x é %i, y é %i\n", x, y);
    trocar(&x, &y);
    printf("x é %i, y é %i\n", x, y);
}

void trocar(int *a, int *b)
{
    int tmp = *a;
    *a = *b;
    *b = tmp;
}
```