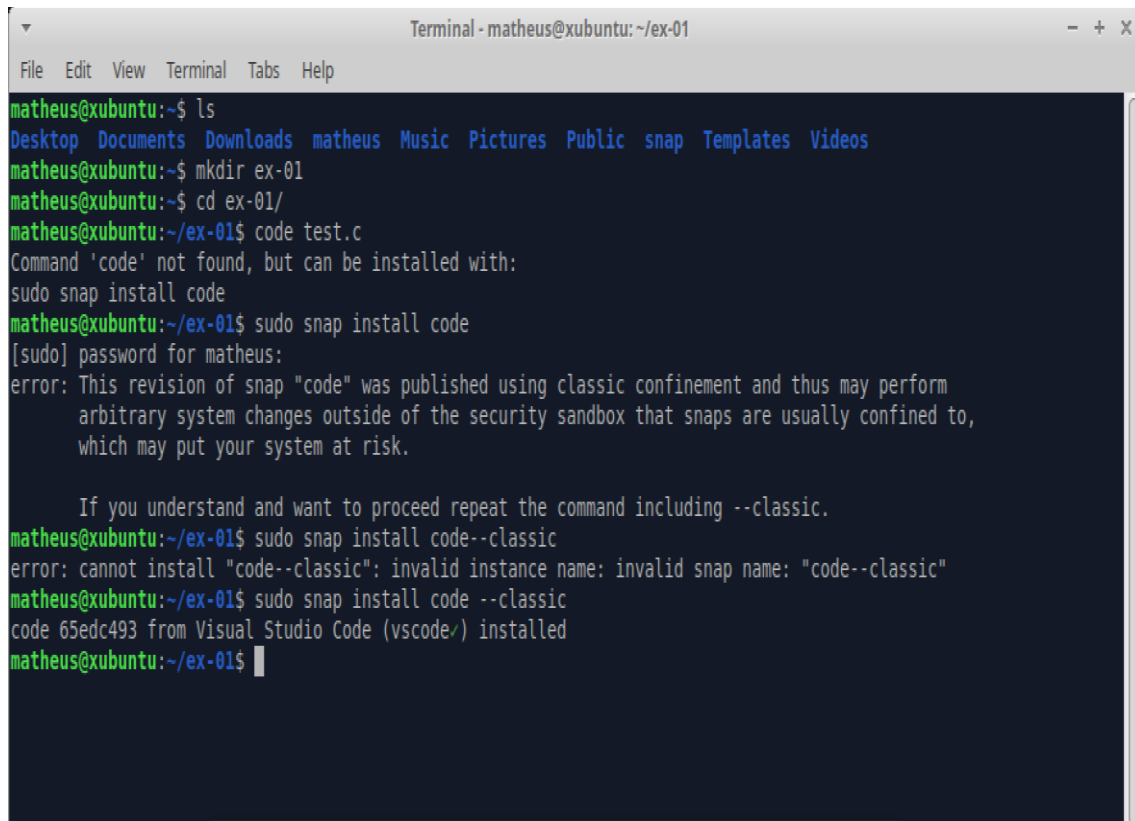


## Exercício 2.

### Criando diretório ex-01



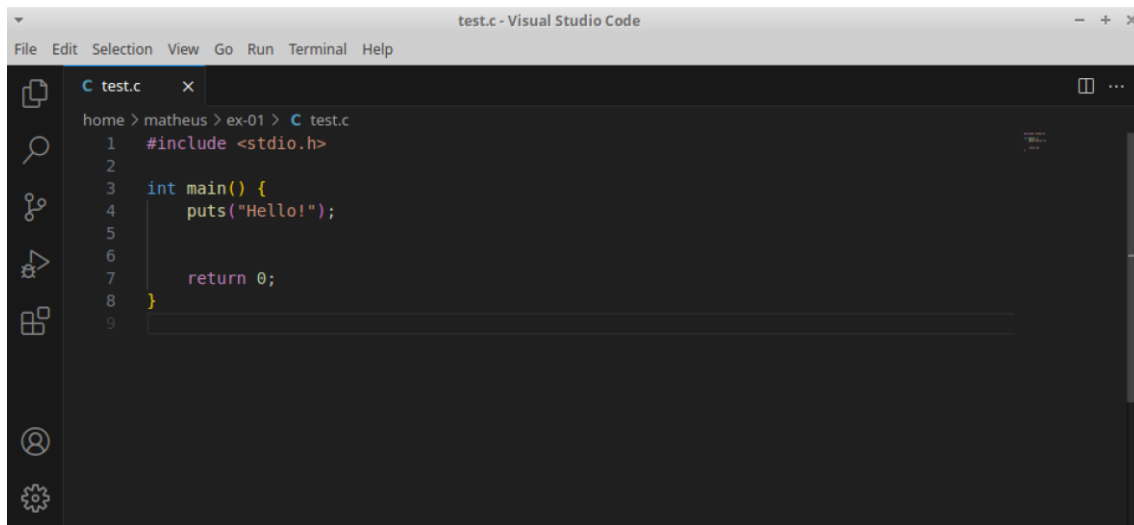
```
Terminal - matheus@xubuntu: ~/ex-01
File Edit View Terminal Tabs Help
matheus@xubuntu:~$ ls
Desktop Documents Downloads matheus Music Pictures Public snap Templates Videos
matheus@xubuntu:~$ mkdir ex-01
matheus@xubuntu:~$ cd ex-01/
matheus@xubuntu:~/ex-01$ code test.c
Command 'code' not found, but can be installed with:
sudo snap install code
matheus@xubuntu:~/ex-01$ sudo snap install code
[sudo] password for matheus:
error: This revision of snap "code" was published using classic confinement and thus may perform
arbitrary system changes outside of the security sandbox that snaps are usually confined to,
which may put your system at risk.

If you understand and want to proceed repeat the command including --classic.
matheus@xubuntu:~/ex-01$ sudo snap install code--classic
error: cannot install "code--classic": invalid instance name: invalid snap name: "code--classic"
matheus@xubuntu:~/ex-01$ sudo snap install code --classic
code 65edc493 from Visual Studio Code (vscode✓) installed
matheus@xubuntu:~/ex-01$
```

\*necessária instalação do Visual Studio Code (“sudo snap install code --classic”)

```
Terminal - matheus@xubuntu: ~/ex-01
File Edit View Terminal Tabs Help
matheus@xubuntu:~$ ls
Desktop Documents Downloads ex-01 matheus Music Pictures Public snap Templates Videos
matheus@xubuntu:~$ cd ex-01/
matheus@xubuntu:~/ex-01$ code test.c
matheus@xubuntu:~/ex-01$
```

Dentro do VS Code: (code test.c)

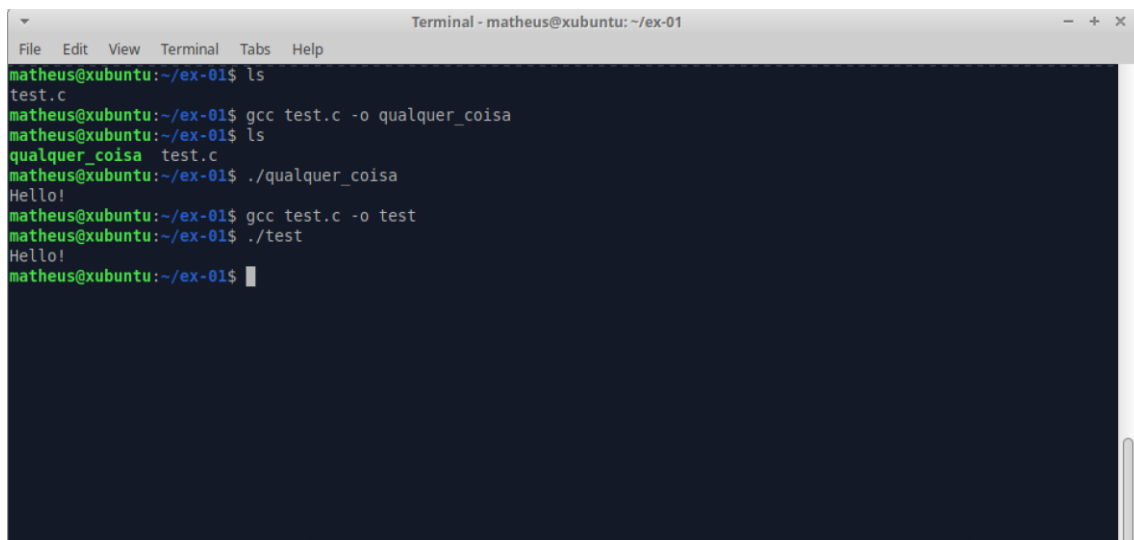


The screenshot shows the Visual Studio Code interface with a file named `test.c` open. The code is a simple C program that prints "Hello!". The editor has a dark theme and a sidebar on the left with icons for Explorer, Search, Source Control, Run and Debug, Extensions, and Settings. The menu bar at the top includes File, Edit, Selection, View, Go, Run, Terminal, and Help.

```
test.c - Visual Studio Code
File Edit Selection View Go Run Terminal Help

1  #include <stdio.h>
2
3  int main() {
4      puts("Hello!");
5
6
7      return 0;
8  }
9
```

Compilando (gcc)

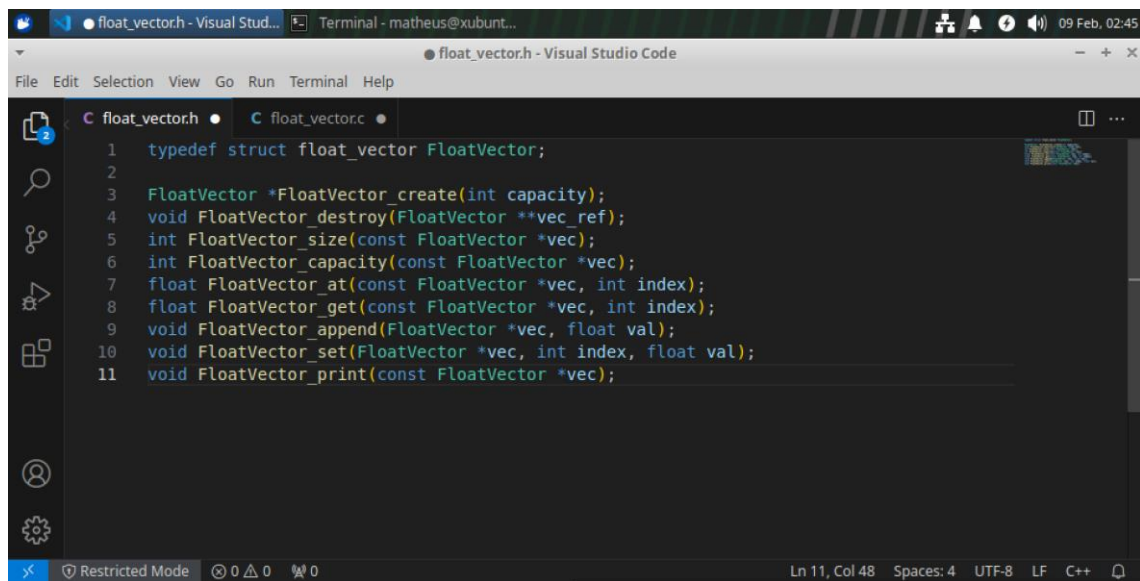


The screenshot shows a terminal window titled "Terminal - matheus@xubuntu: ~/ex-01". It displays the commands used to compile and run the C program. The output shows the file `test.c` being listed, compiled into `qualquer_coisa`, and then executed to produce the output "Hello!".

```
Terminal - matheus@xubuntu: ~/ex-01
File Edit View Terminal Tabs Help

matheus@xubuntu:~/ex-01$ ls
test.c
matheus@xubuntu:~/ex-01$ gcc test.c -o qualquer_coisa
matheus@xubuntu:~/ex-01$ ls
qualquer_coisa  test.c
matheus@xubuntu:~/ex-01$ ./qualquer_coisa
Hello!
matheus@xubuntu:~/ex-01$ gcc test.c -o test
matheus@xubuntu:~/ex-01$ ./test
Hello!
matheus@xubuntu:~/ex-01$
```

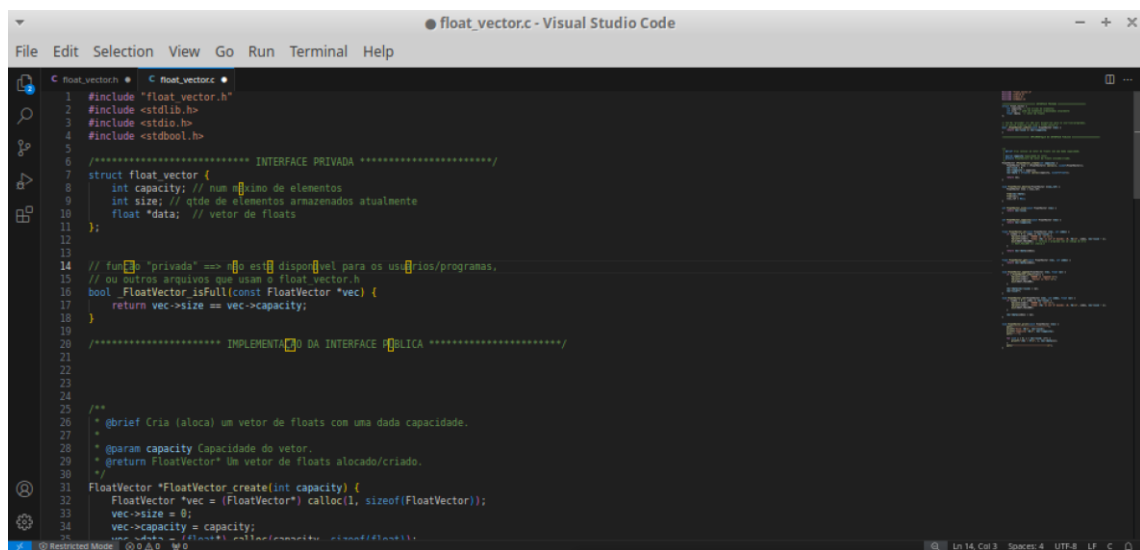
## .float\_vector.h



The screenshot shows the Visual Studio Code editor with the file `float_vector.h` open. The code defines a `FloatVector` struct and a set of functions to manage it. The status bar at the bottom indicates the cursor is at line 11, column 48.

```
1 typedef struct float_vector FloatVector;
2
3 FloatVector *FloatVector_create(int capacity);
4 void FloatVector_destroy(FloatVector **vec_ref);
5 int FloatVector_size(const FloatVector *vec);
6 int FloatVector_capacity(const FloatVector *vec);
7 float FloatVector_at(const FloatVector *vec, int index);
8 float FloatVector_get(const FloatVector *vec, int index);
9 void FloatVector_append(FloatVector *vec, float val);
10 void FloatVector_set(FloatVector *vec, int index, float val);
11 void FloatVector_print(const FloatVector *vec);
```

## .float\_vector.c



The screenshot shows the Visual Studio Code editor with the file `float_vector.c` open. The code implements the functions declared in the header file. It includes standard library headers, defines a private interface for the struct, and implements the `FloatVector_create` function. The status bar at the bottom indicates the cursor is at line 14, column 3.

```
1 #include "float_vector.h"
2 #include <stdlib.h>
3 #include <stdio.h>
4 #include <stdbool.h>
5
6 /***** INTERFACE PRIVADA *****/
7 struct float_vector {
8     int capacity; // número de elementos
9     int size; // qtd de elementos armazenados atualmente
10    float *data; // vetor de floats
11 };
12
13
14 // função "privada" => não está disponível para os usuários/programas,
15 // ou outros arquivos que usam o float_vector.h
16 bool FloatVector_isFull(const FloatVector *vec) {
17     return vec->size == vec->capacity;
18 }
19
20 /***** IMPLEMENTAÇÃO DA INTERFACE PÚBLICA *****/
21
22
23 /**
24  * @brief Cria (aloca) um vetor de floats com uma dada capacidade.
25  *
26  * @param capacity Capacidade do vetor.
27  * @return FloatVector* Um vetor de floats alocado/criado.
28  */
29 FloatVector *FloatVector_create(int capacity) {
30     FloatVector *vec = (FloatVector*) calloc(1, sizeof(FloatVector));
31     vec->size = 0;
32     vec->capacity = capacity;
33     vec->data = (float*) malloc(sizeof(float) * vec->capacity);
```

## Compilação tipo abstrato de dados

```
Terminal - matheus@xubuntu: ~/ex-02
File Edit View Terminal Tabs Help
matheus@xubuntu:~/ex-02$ ls
float_vector.c float_vector.h
matheus@xubuntu:~/ex-02$ gcc -c float_vector.c
matheus@xubuntu:~/ex-02$ ls
float_vector.c float_vector.h float_vector.o
matheus@xubuntu:~/ex-02$
```

```
Terminal - matheus@xubuntu: ~/ex-02
File Edit View Terminal Tabs Help
matheus@xubuntu:~/ex-02$ mkdir obj
matheus@xubuntu:~/ex-02$ ls
float_vector.c float_vector.h float_vector.o obj
matheus@xubuntu:~/ex-02$ gcc -c float_vector.c -o obj/float_vector.o
matheus@xubuntu:~/ex-02$ ls obj/
float_vector.o
matheus@xubuntu:~/ex-02$ tree .
.
├── float_vector.c
├── float_vector.h
├── float_vector.o
└── obj
    └── float_vector.o

2 directories, 4 files
matheus@xubuntu:~/ex-02$
```