

Typedef void*

Teoria e Implementação

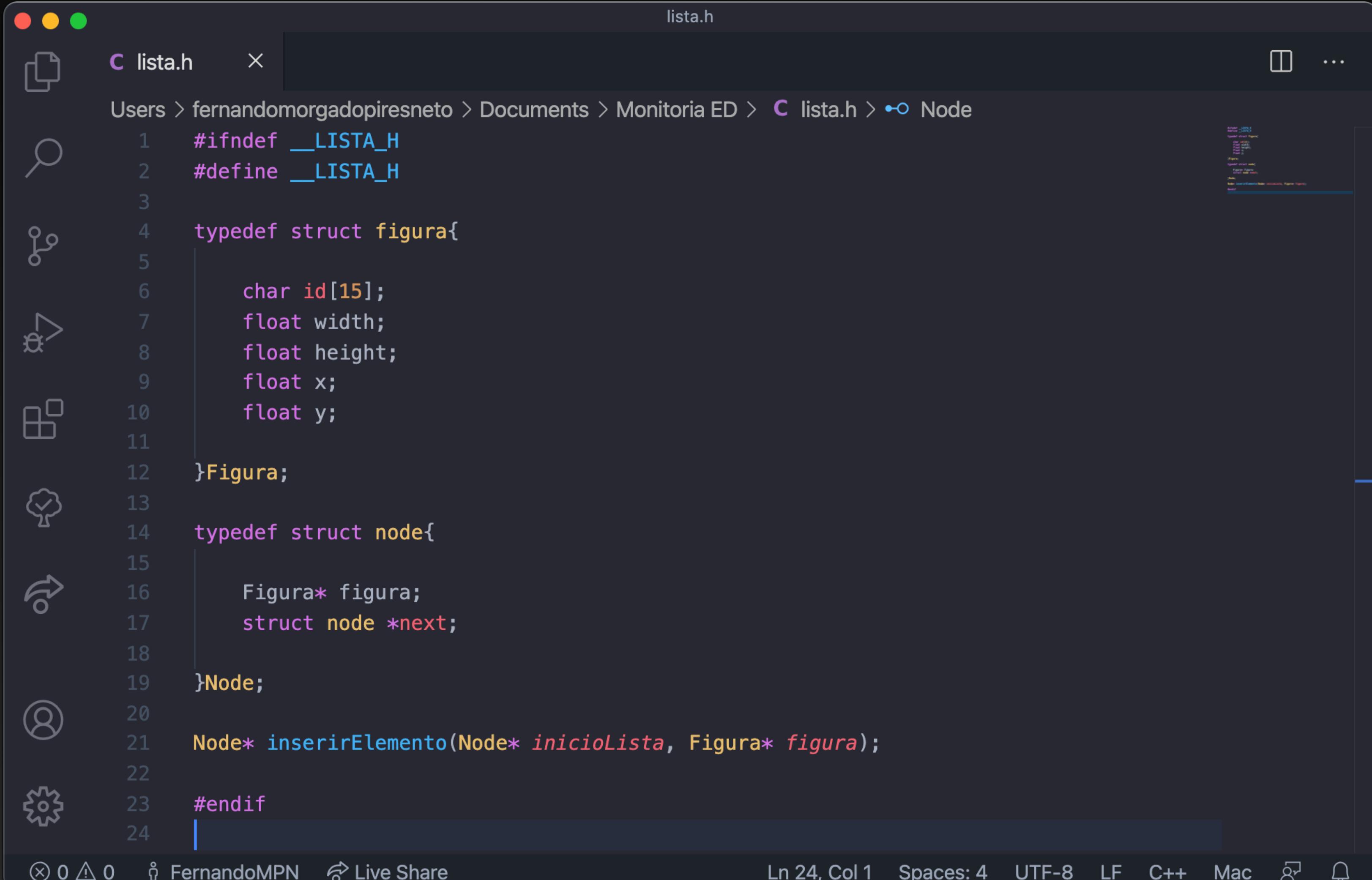
Fernando Morgado Pires Neto

Anteriormente no T1...

Como era feita a organização

- As structs que representavam as figuras estavam no .h
- A estrutura de dado implementada "sabia" qual era a figura que ia ser inserida

Como era feita a organização



The screenshot shows a dark-themed code editor window titled "lista.h". The file path is "Users > fernandomorgadopiresneto > Documents > Monitoria ED > lista.h". The code defines two structures: "Figura" and "Node". The "Figura" structure contains fields for an ID string and three floating-point numbers (width, height, x, y). The "Node" structure contains a pointer to a "Figura" object and a pointer to the next node in a linked list. A function prototype "Node* inserirElemento(Node* inicioLista, Figura* figura);" is also present. The code is written in C, using "#ifndef", "#define", and "#endif" directives.

```
#ifndef __LISTA_H
#define __LISTA_H

typedef struct figura{
    char id[15];
    float width;
    float height;
    float x;
    float y;
}Figura;

typedef struct node{
    Figura* figura;
    struct node *next;
}Node;

Node* inserirElemento(Node* inicioLista, Figura* figura);

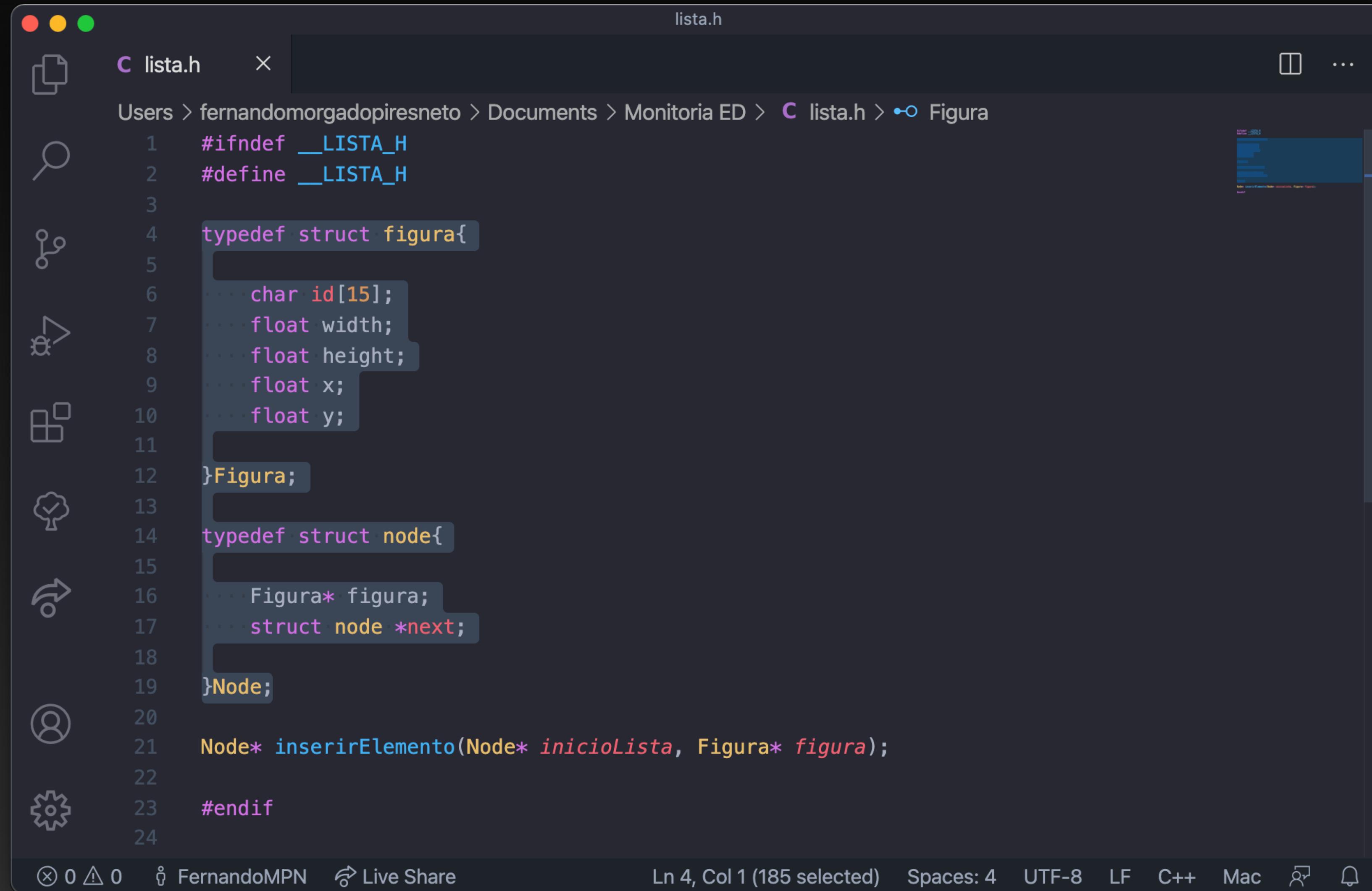
#endif
```

Agora...

Como deve ser feita a organização

- As structs **não** podem mais ir nos .h, **apenas** no .c
- Os .h dos TADs são reservados apenas para mostrar a **assinatura** dos métodos que serão exportados.

Como deve ser feita a organização

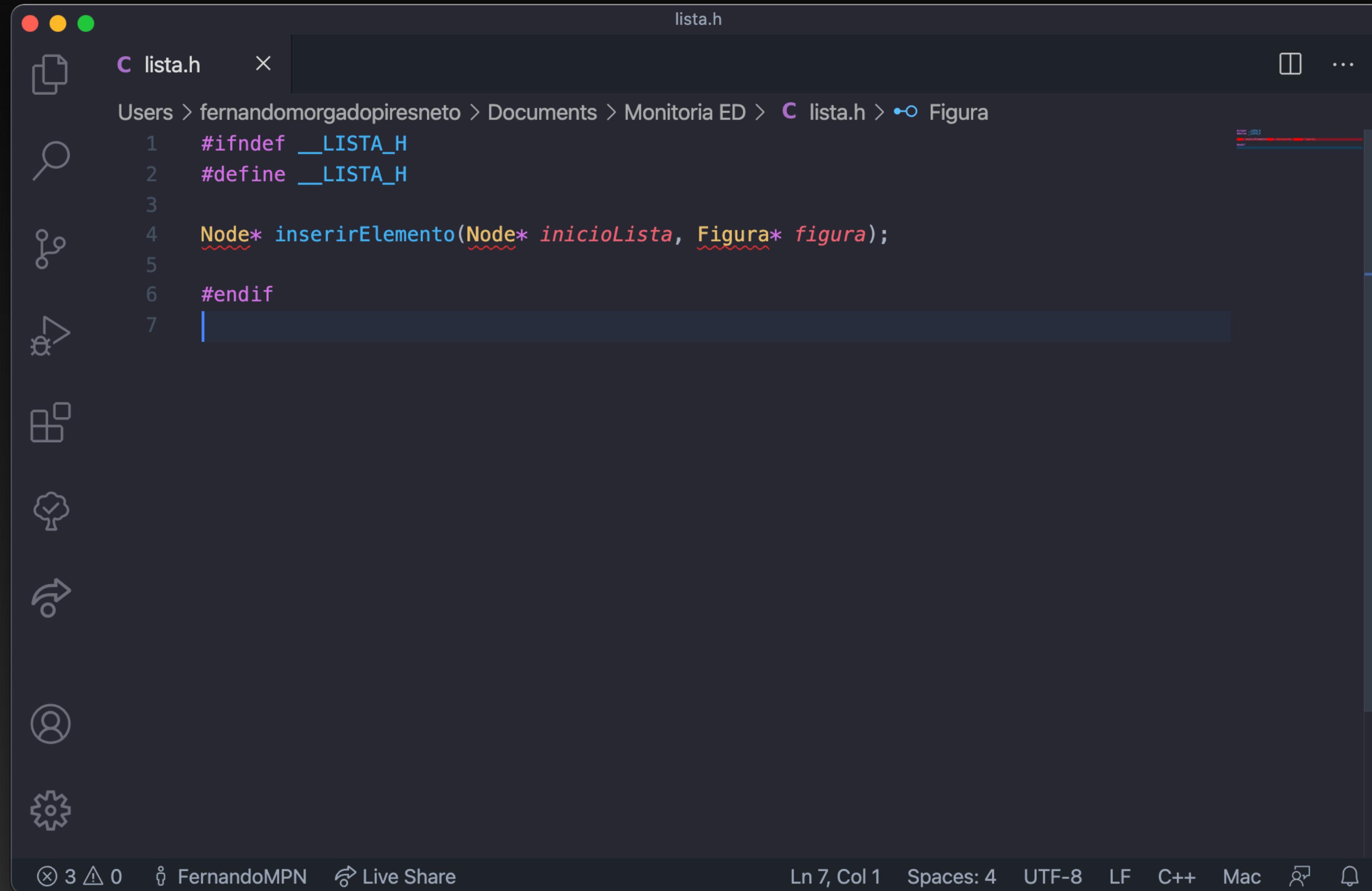


The screenshot shows a dark-themed code editor window titled "lista.h". The file path is "Users > fernandomorgadopiresneto > Documents > Monitoria ED > lista.h". The code is a C header file defining two structures: "figura" and "node". The "figura" structure contains fields for an ID string, width, height, x, and y coordinates. The "node" structure contains a pointer to a "figura" object and a pointer to the next node in a linked list. The file also includes a macro definition for "LISTA_H" and a function prototype for inserting elements into a list.

```
1 #ifndef __LISTA_H
2 #define __LISTA_H
3
4 typedef struct figura{
5     ...
6     char id[15];
7     float width;
8     float height;
9     float x;
10    float y;
11 }
12 }Figura;
13
14 typedef struct node{
15     ...
16     Figura* figura;
17     struct node *next;
18 }
19 }Node;
20
21 Node* inserirElemento(Node* inicioLista, Figura* figura);
22
23#endif
```

Bottom status bar: ⌘ 0 ⌛ 0 ⌂ FernandoMPN ⌂ Live Share Ln 4, Col 1 (185 selected) Spaces: 4 UTF-8 LF C++ Mac ⌂ ⌂

Como deve ser feita a organização

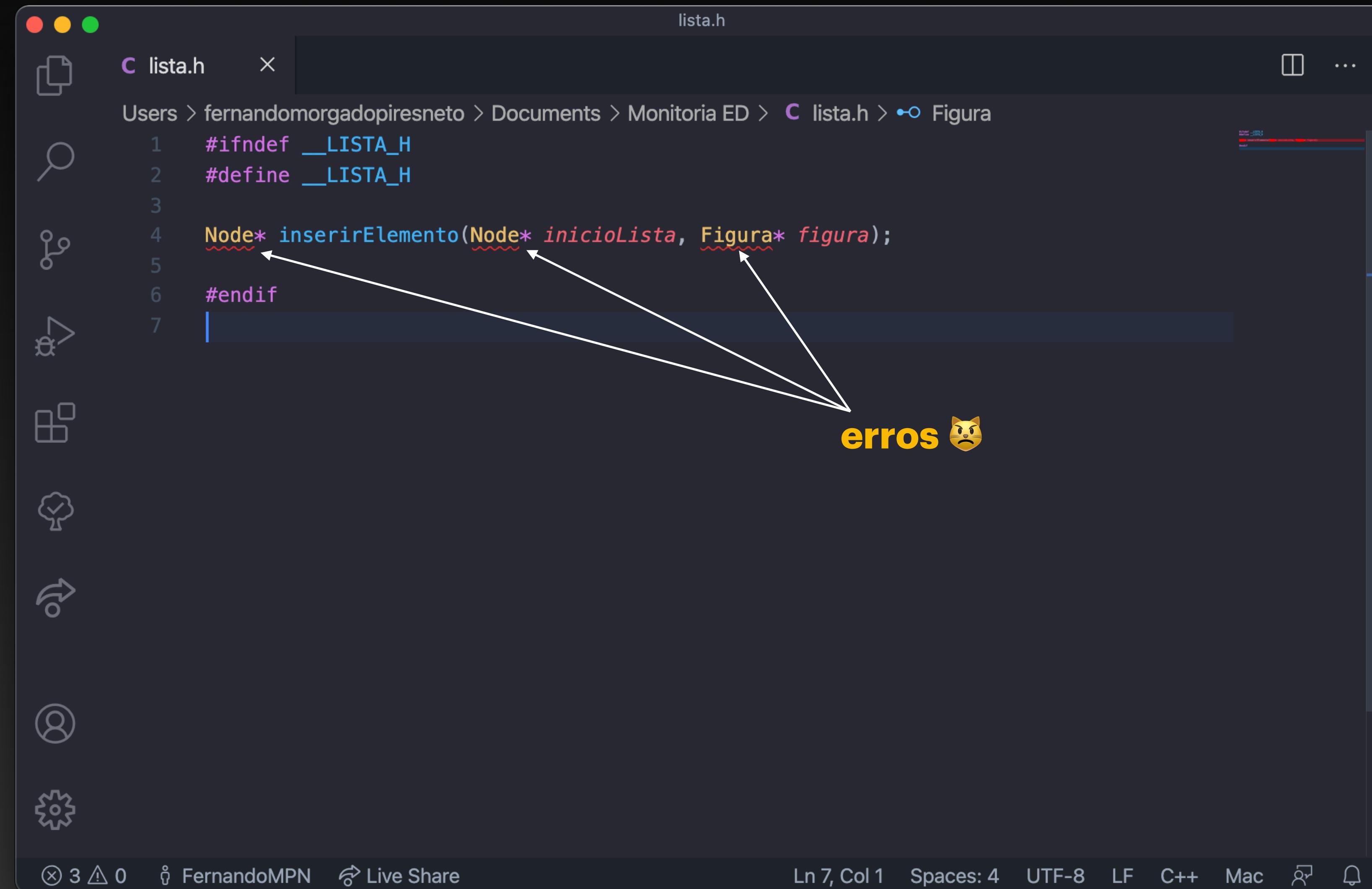


The screenshot shows a dark-themed code editor window titled "lista.h". The file path is "Users > fernandomorgadopiresneto > Documents > Monitoria ED > lista.h". The code content is as follows:

```
1 #ifndef __LISTA_H
2 #define __LISTA_H
3
4 Node* inserirElemento(Node* inicioLista, Figura* figura);
5
6 #endif
7
```

The code editor has a sidebar on the left with various icons for file operations like copy, paste, search, and refresh. The status bar at the bottom shows "Ln 7, Col 1" and "Spaces: 4".

Como deve ser feita a organização



The screenshot shows a dark-themed code editor window titled "lista.h". The file path is "Users > fernandomorgadopiresneto > Documents > Monitoria ED > lista.h". The code contains the following lines:

```
1 #ifndef __LISTA_H
2 #define __LISTA_H
3
4 Node* inserirElemento(Node* inicioLista, Figura* figura);
5
6#endif
7
```

Annotations with arrows point from the text "erros 😠" to the identifiers "Node*", "inicioLista", and "Figura*". The status bar at the bottom shows "Ln 7, Col 1" and "Spaces: 4".

Como deve ser feita a organização

- Ao incluir as structs no .c de seus respectivos TADs, os .h **não** tem como saber qual tipo a função está referenciando.

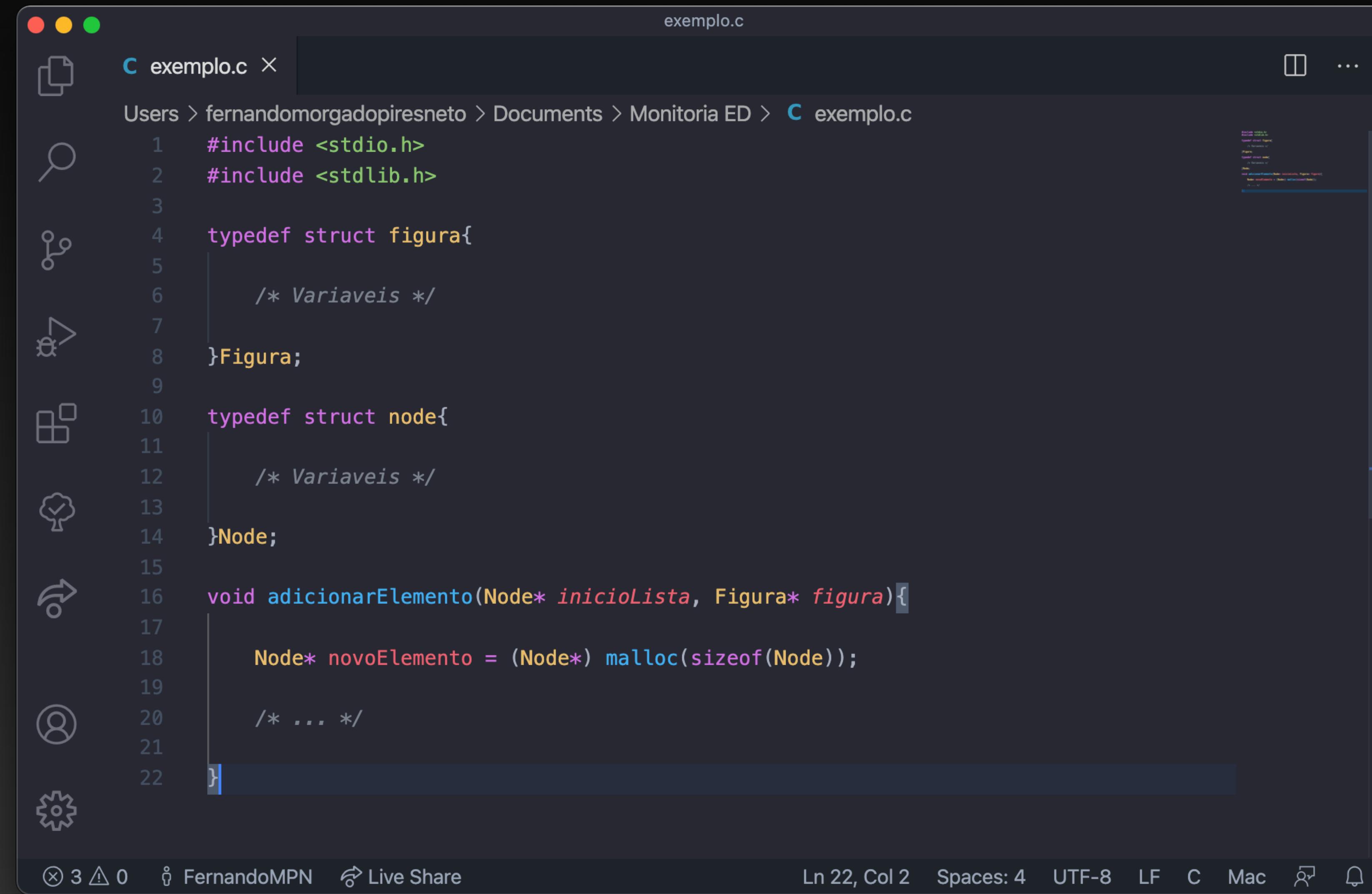
Relembrando alguns conceitos

- O que é um ponteiro?

Relembrando alguns conceitos

- O que é um ponteiro?
 - Aponta para o endereço de memória armazenado pela variável.

Relembrando alguns conceitos



The screenshot shows a dark-themed code editor window titled "exemplo.c". The file path is "Users > fernandomorgadopiresneto > Documents > Monitoria ED > exemplo.c". The code implements a linked list structure:

```
#include <stdio.h>
#include <stdlib.h>

typedef struct figura{
    /* Variaveis */
}Figura;

typedef struct node{
    /* Variaveis */
}Node;

void adicionarElemento(Node* inicioLista, Figura* figura){
    Node* novoElemento = (Node*) malloc(sizeof(Node));
    /* ... */
}
```

The editor interface includes a sidebar with icons for file operations like new, open, save, and search. The bottom status bar shows file statistics: 3 lines, 0 changes, author "FernandoMPN", and "Live Share". It also displays the current line (Ln 22), column (Col 2), spaces (Spaces: 4), and encoding (UTF-8). The status bar also includes buttons for LF, C, Mac, and a gear icon.

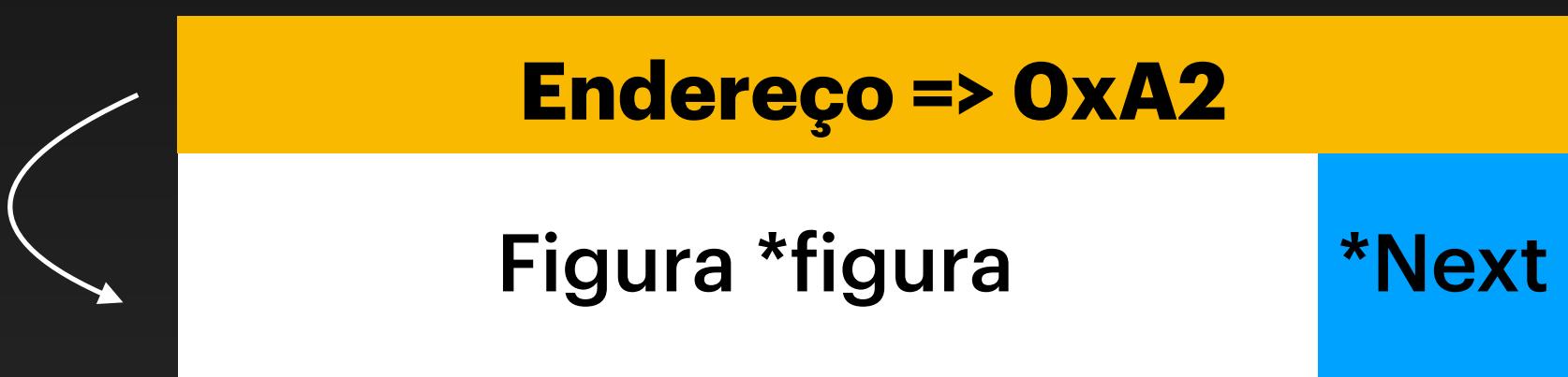
Void*

Ponteiro void (`void*`)

- É um tipo de variável que **não possui tipo definido**.
- Apenas aponta para o endereço de memória que sua variável armazena.

Ponteiro void (void*)

```
Node* novoElemento = (Node*) malloc(sizeof(Node));
```

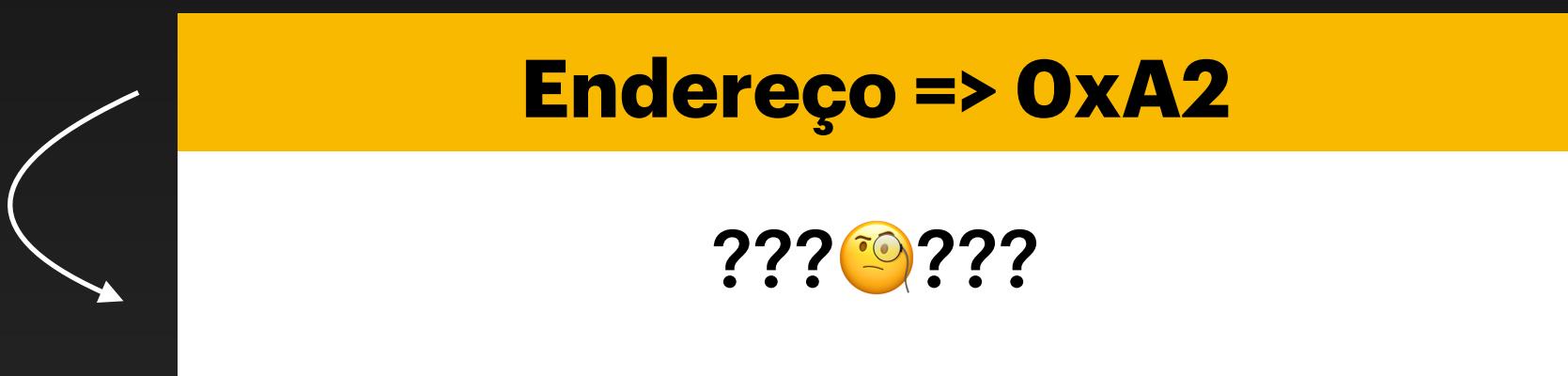


Node* novoElemento

Elemento da lista alocado dinamicamente

Ponteiro void (void*)

```
void* novoElemento = (void*) malloc(sizeof(Node));
```



void* novoElemento

Elemento da lista alocado dinamicamente

Ponteiro void (`void*`)

- Quando vamos alterar o conteúdo apontado por uma variável do tipo `void*` é preciso **indicar** qual é o tipo original dela.

Ponteiro void (`void*`)

- Quando vamos alterar o conteúdo apontado por uma variável do tipo `void*` é preciso **indicar** qual é o tipo original dela.
- Para isso, vamos utilizar o **casting**.

Casting

Endereço => 0xA2

???????

void*



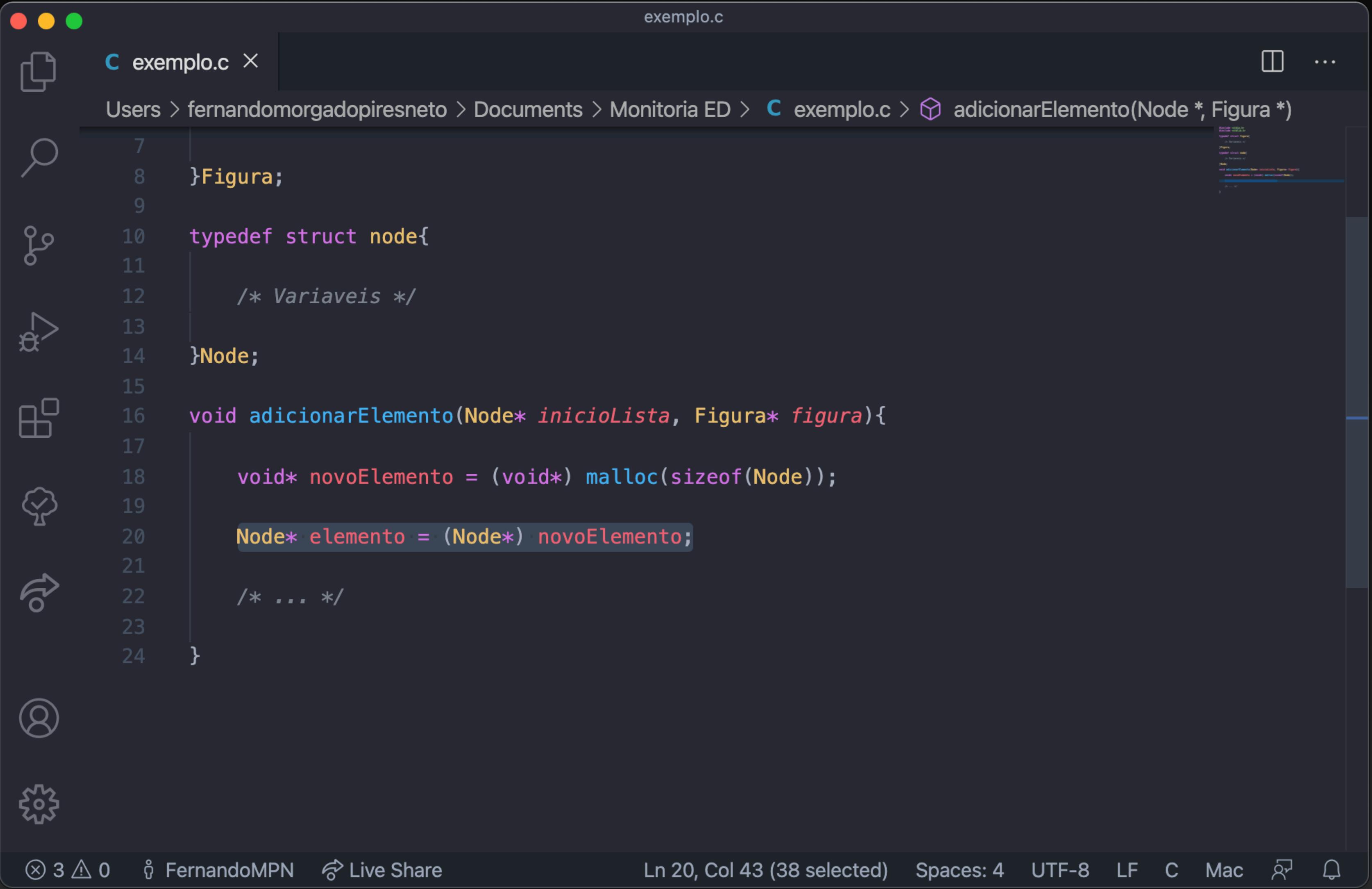
Endereço => 0xA2

Figura *figura

*Next

Node*

Casting



The screenshot shows a dark-themed code editor window titled "exemplo.c". The file path is "Users > fernandomorgadopiresneto > Documents > Monitoria ED > exemplo.c". The code is as follows:

```
7 }Figura;
8
9
10 typedef struct node{
11
12     /* Variaveis */
13
14 }Node;
15
16 void adicionarElemento(Node* inicioLista, Figura* figura){
17
18     void* novoElemento = (void*) malloc(sizeof(Node));
19
20     Node* elemento = (Node*) novoElemento;
21
22     /* ... */
23
24 }
```

The line "Node* elemento = (Node*) novoElemento;" is highlighted with a blue selection bar. A tooltip or context menu is visible on the right side of the editor, listing options like "Formatar seleção", "Formatar linha", "Formatar bloco", "Formatar todo o documento", "Formatar todo o arquivo", and "Formatar todo o projeto".

At the bottom of the editor, there are status indicators: "Ln 20, Col 43 (38 selected)", "Spaces: 4", "UTF-8", "LF", "C", "Mac", and icons for "Live Share", "Bell", and other settings.

Ponteiro void (void*)

- Criar variáveis void* dentro próprio arquivo original não faz muito sentido.

Ponteiro void (void*)

- Criar variáveis void* dentro próprio arquivo original não faz muito sentido.
- A ideia é poder **armazenar** os endereços alocados em outros arquivos, e poder utilizá-los em seu arquivo original.

Ponteiro void (`void*`)

main.c

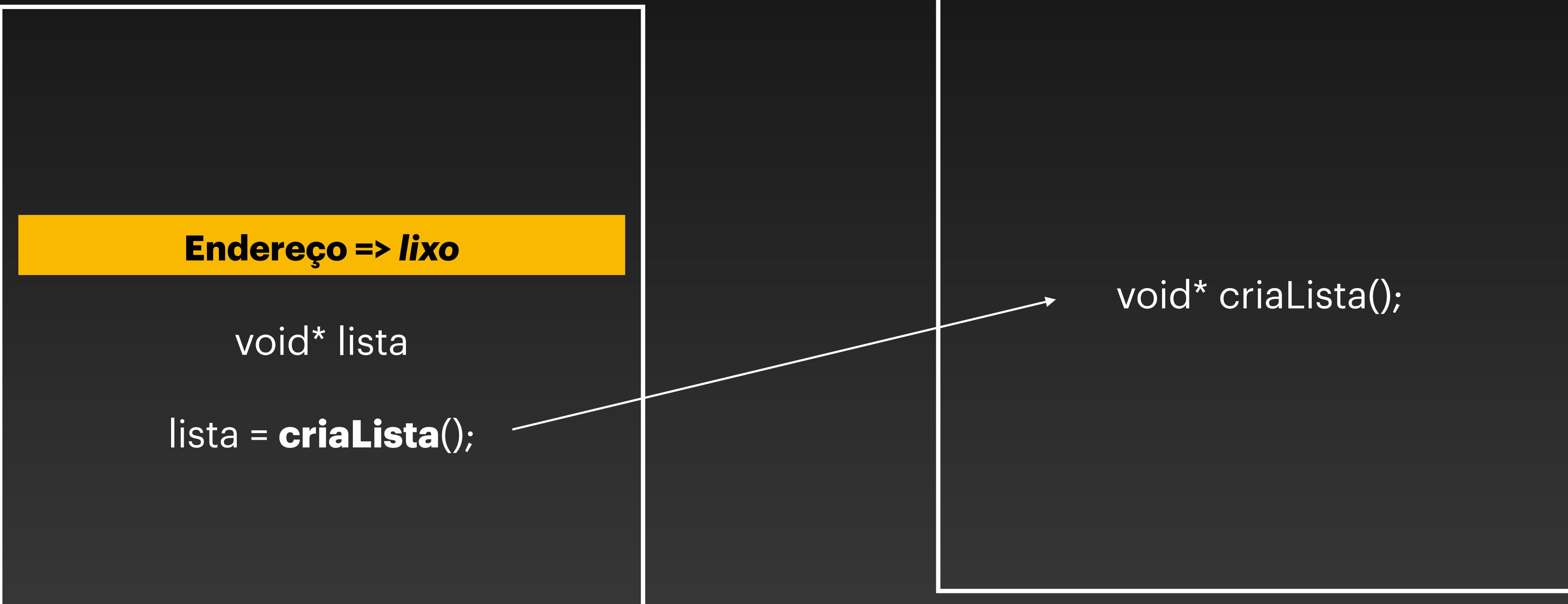
Endereço => lixo

`void* lista`

`lista = criaLista();`

lista.c

`void* criaLista();`



Ponteiro void (void*)

main.c

Endereço => lixo

void* lista

lista = criaLista();

lista.c

void* criaLista();

Endereço => 0xA2

Figura *figura

*Next

Ponteiro void (void*)

main.c

Endereço => 0xA2

??? 🙄 ???

void* lista

lista = criaLista();

lista.c

void* criaLista();

Endereço => 0xA2

Figura *figura

*Next

Retorno



Ponteiro void (void*)

main.c

```
Endereço => 0xA2  
??? 🙄 ???  
  
void* lista  
  
lista = adicionaElemento(void* lista,  
                           void* figura);
```

lista.c

```
void* adicionaElemento(void* lista,  
                      void* figura);
```

Ponteiro void (void*)

main.c

```
Endereço => 0xA2  
??? 🙄 ???  
  
void* lista  
  
lista = adicionaElemento(void* lista,  
                           void* figura);
```

lista.c

```
Endereço => 0xA2  
??? 🙄 ???  
  
void* adicionaElemento(void* lista,  
                      void* figura);
```

Ponteiro void (void*)

main.c

```
Endereço => 0xA2
??? 😐 ???

void* lista

lista = adicionaElemento(void* lista,
                        void* figura);
```

lista.c

```
Endereço => 0xA2
??? 😐 ???

void* adicionaElemento(void* lista,
                      void* figura);

>>CASTING<<
```

Ponteiro void (void*)

main.c

```
Endereço => 0xA2  
??? 🙄 ???  
  
void* lista  
  
lista = adicionaElemento(void* lista,  
                        void* figura);
```

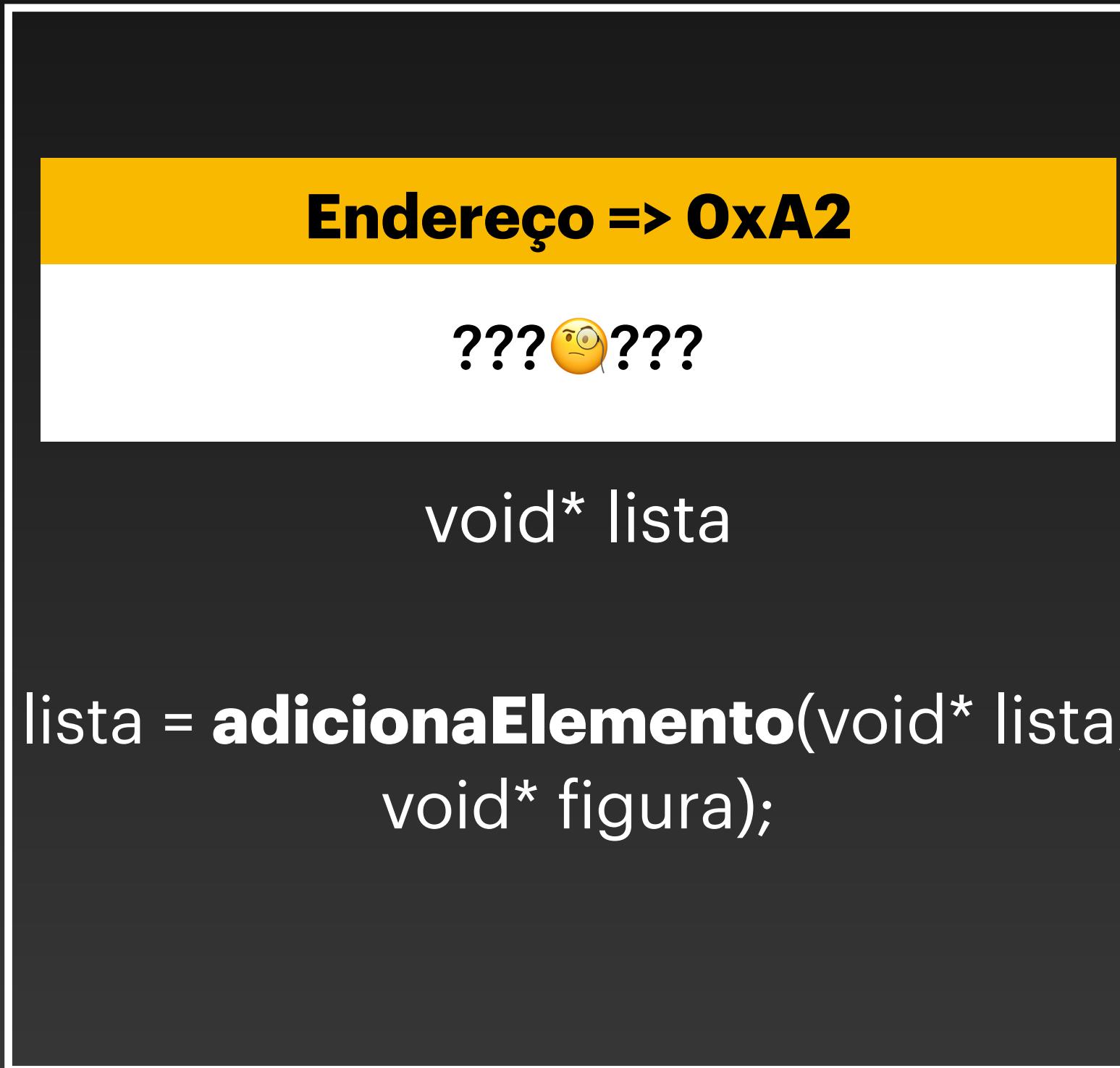
lista.c

```
void* adicionaElemento(void* lista,  
                      void* figura);
```

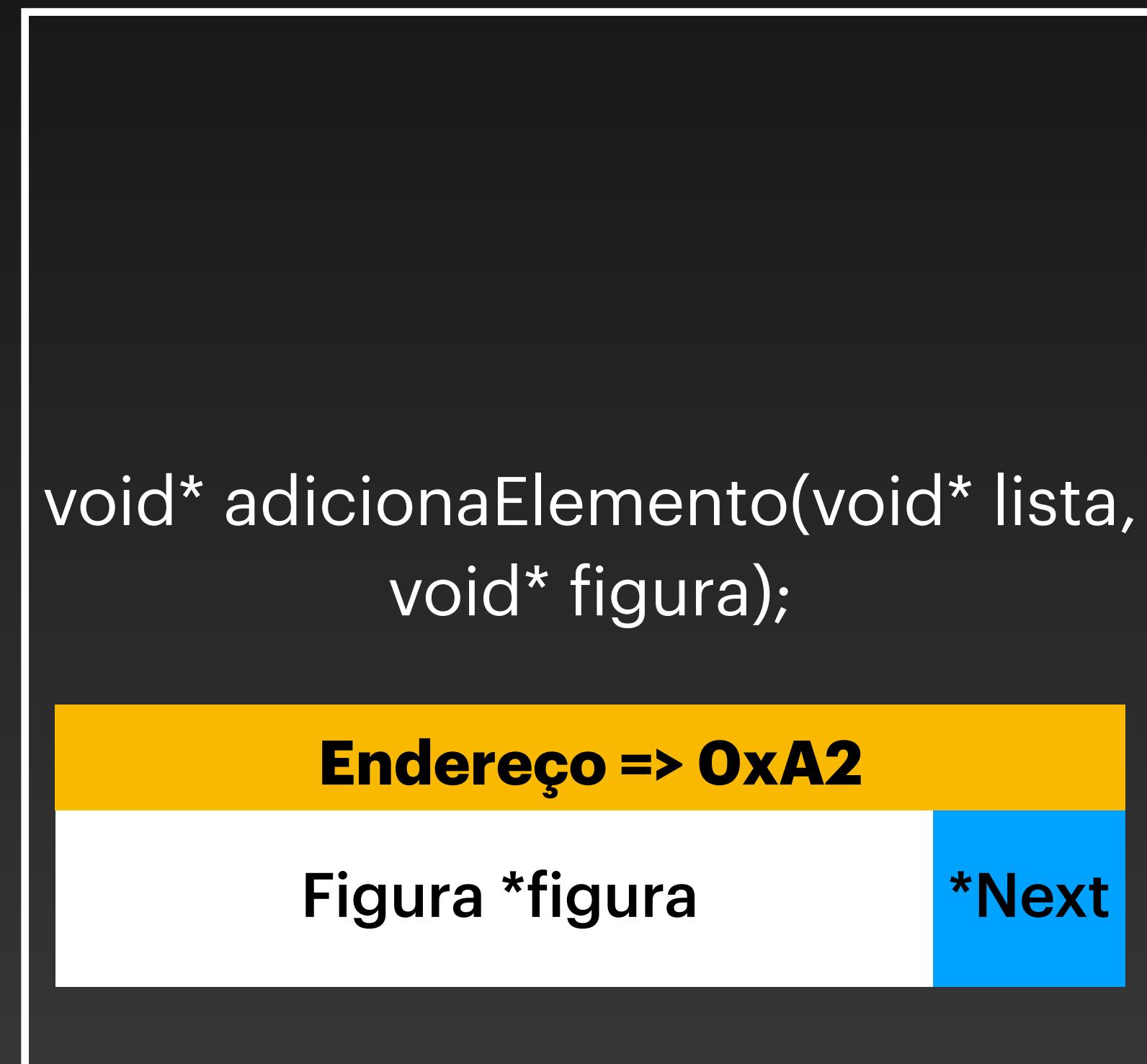
Endereço => 0xA2
Figura *figura *Next

Ponteiro void (void*)

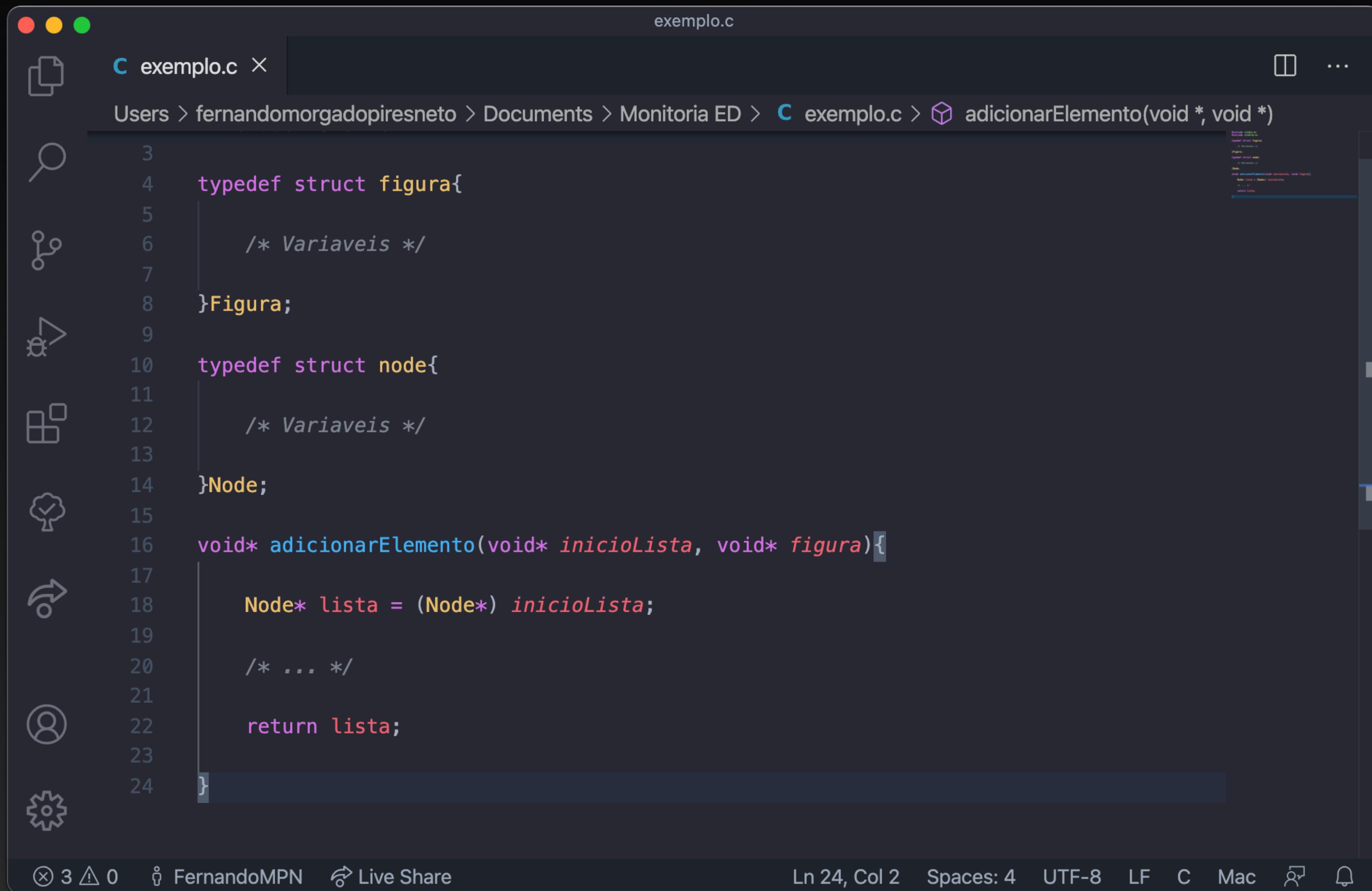
main.c



lista.c



Ponteiro void (void*)

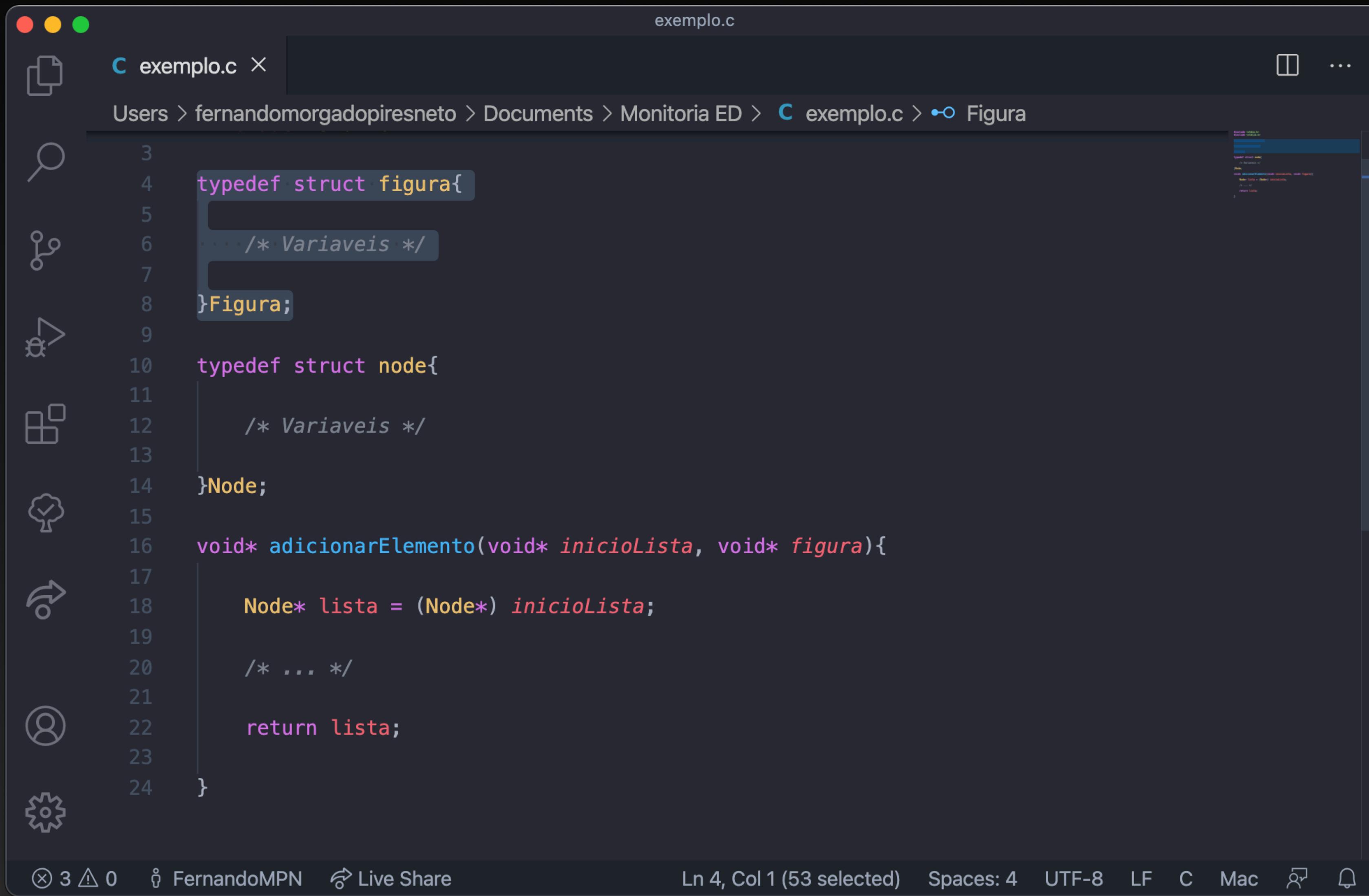


The screenshot shows a dark-themed code editor window titled "exemplo.c". The file path is "Users > fernandomorgadopiresneto > Documents > Monitoria ED > exemplo.c". The code defines two structures: "figura" and "node", and a function "adicionarElemento".

```
3
4  typedef struct figura{
5
6      /* Variaveis */
7
8  }Figura;
9
10 typedef struct node{
11
12     /* Variaveis */
13
14 }Node;
15
16 void* adicionarElemento(void* inicioLista, void* figura){
17
18     Node* lista = (Node*) inicioLista;
19
20     /* ... */
21
22     return lista;
23
24 }
```

The code editor includes standard icons for file operations (New, Open, Save, Find, Copy, Paste, etc.) and a status bar at the bottom with file statistics: Line 24, Column 2, Spaces: 4, UTF-8, LF, C, Mac, and a Live Share icon.

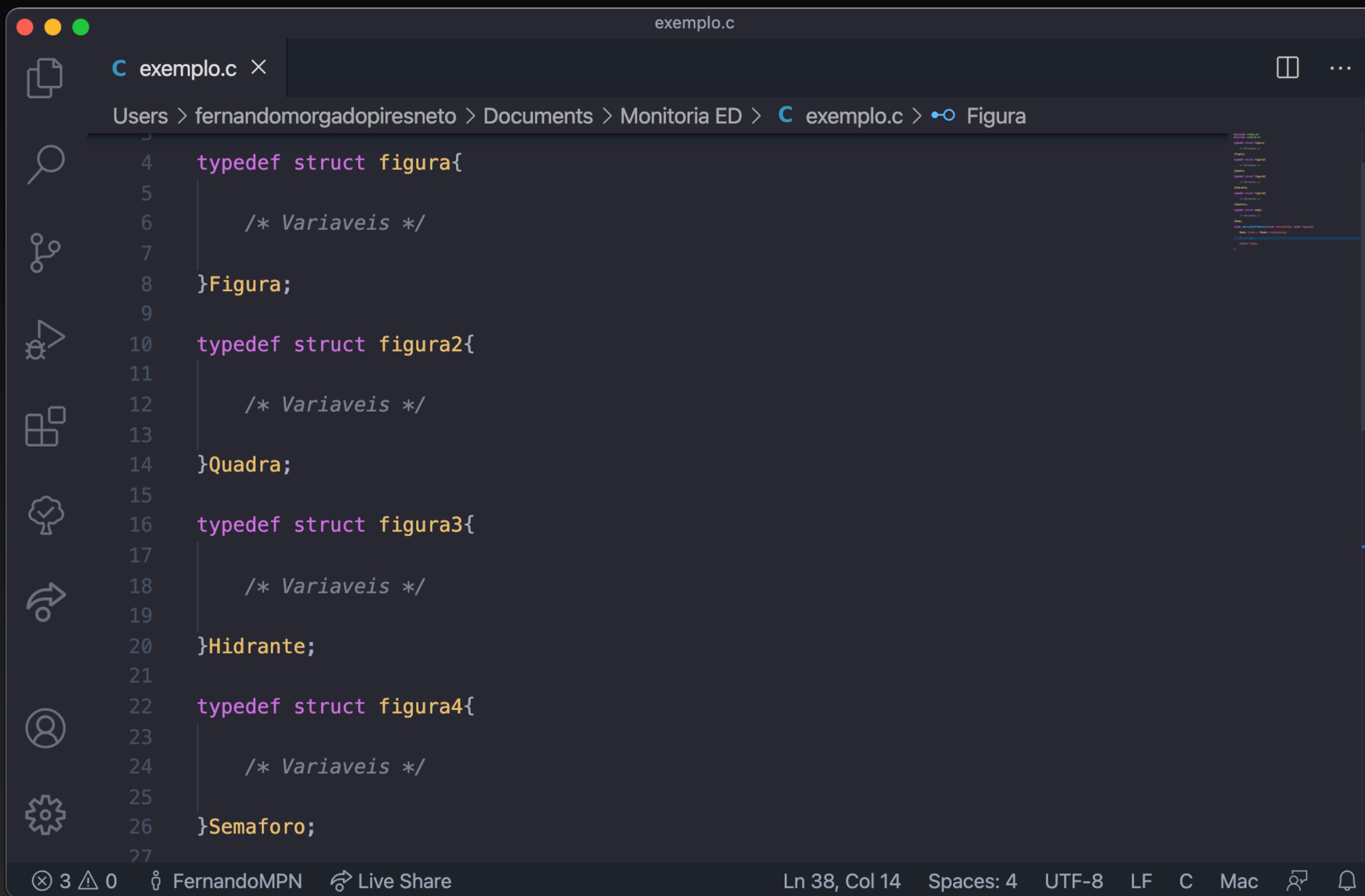
Ponteiro void (void*)



```
exemplo.c
C exemplo.c ×
Users > fernandomorgadopiresneto > Documents > Monitoria ED > C exemplo.c > Figura
3
4 typedef struct figura{
5
6     /* Variaveis */
7
8 }Figura;
9
10 typedef struct node{
11
12     /* Variaveis */
13
14 }Node;
15
16 void* adicionarElemento(void* inicioLista, void* figura){
17
18     Node* lista = (Node*) inicioLista;
19
20     /* ... */
21
22     return lista;
23
24 }
```

Bottom status bar: ⌂ 3 ⚠ 0 ⌂ FernandoMPN ⌂ Live Share Ln 4, Col 1 (53 selected) Spaces: 4 UTF-8 LF C Mac ⌂ ⌂

Ponteiro void (void*)

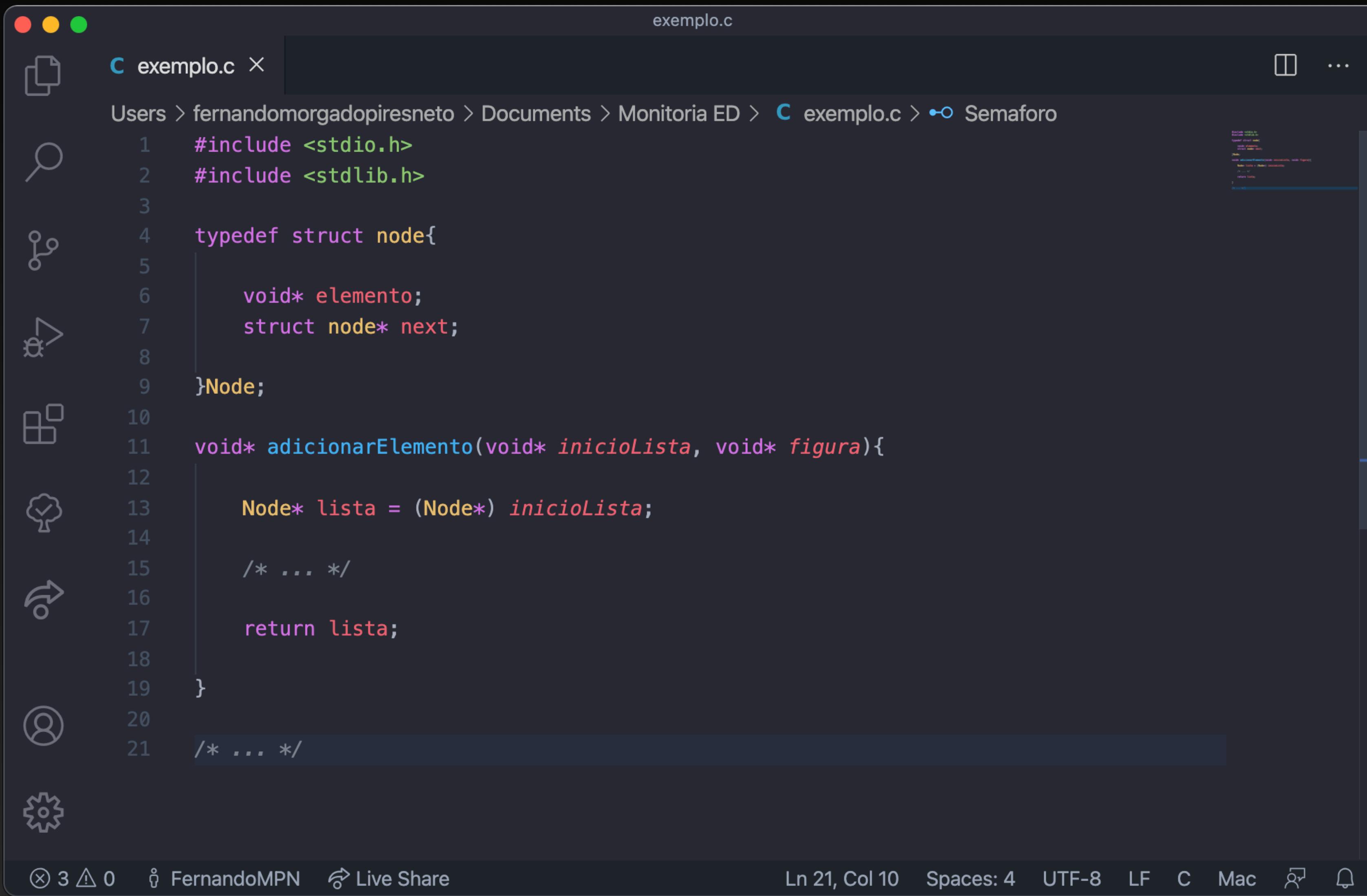


The screenshot shows a dark-themed code editor window titled "exemplo.c". The file path is "Users > fernandomorgadopiresneto > Documents > Monitoria ED > C exemplo.c > Figura". The code defines four structures:

```
4  typedef struct figura{  
5      /* Variaveis */  
6  }Figura;  
7  
10  typedef struct figura2{  
11      /* Variaveis */  
12  }Quadra;  
13  
16  typedef struct figura3{  
17      /* Variaveis */  
18  }Hidrante;  
19  
22  typedef struct figura4{  
23      /* Variaveis */  
24  }Semaforo;  
25  
26  
```

The status bar at the bottom indicates the file has 3 lines, 0 errors, and is owned by FernandoMPN. It also shows "Live Share" integration, line 38, column 14, spaces: 4, and encoding: UTF-8.

Ponteiro void (void*)



The screenshot shows a dark-themed code editor window titled "exemplo.c". The file path is "Users > fernandomorgadopiresneto > Documents > Monitoria ED > exemplo.c". The code defines a linked list node structure and an insertion function:

```
#include <stdio.h>
#include <stdlib.h>

typedef struct node{
    void* elemento;
    struct node* next;
}Node;

void* adicionarElemento(void* inicioLista, void* figura){

    Node* lista = (Node*) inicioLista;

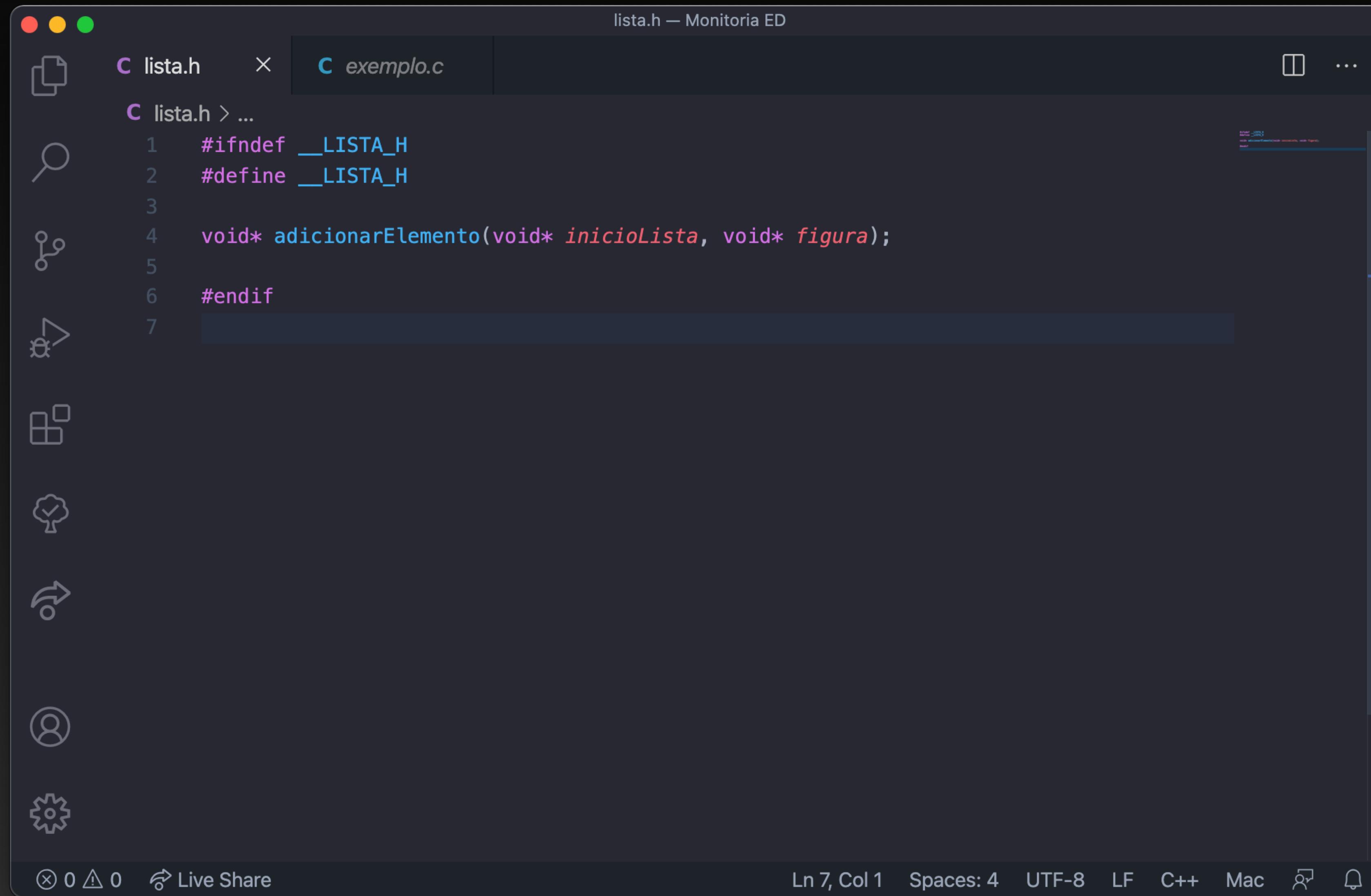
    /* ... */

    return lista;
}

/* ... */
```

The code editor interface includes a sidebar with icons for file operations like new, open, save, and search. The bottom status bar shows file statistics (3 lines, 0 errors), the author ("FernandoMPN"), live share status, line and column numbers (Ln 21, Col 10), character encoding (UTF-8), and file type indicators (LF, C, Mac).

Ponteiro void (void*)



The screenshot shows a dark-themed code editor window titled "lista.h — Monitoria ED". The left sidebar contains icons for file operations, search, navigation, and settings. The main pane displays the following C code:

```
#ifndef __LISTA_H
#define __LISTA_H

void* adicionarElemento(void* inicioLista, void* figura);

#endif
```

The code editor interface includes standard status bar elements at the bottom: line 7, column 1; spaces: 4; encoding: UTF-8; line endings: LF; language: C++; file type: Mac; and a live share icon.

Ponteiro void (`void*`)

- Pronto, agora você já sabe como armazenar e transferir endereços usando ponteiros `void!` 😊🎉🎊

Getters and Setters

Situação

- Você precisa realizar alguma operação **fora** do arquivo original.
- Ex: buscar elemento, alterar valor, consultar valor da variável, ...

Situação

qry.c

Endereço => 0xB5

??? 🙄 ???

quadra.c

Endereço => 0xB5

char cep[] = "abc" float area = 50.0

Situação

qry.c

Endereço => 0xB5

??? 🙄 ???

Qual é a área da quadra?

quadra.c

Endereço => 0xB5

char cep[] = "abc" float area = 50.0

Situação

qry.c

quadra.c

Endereço => 0xB5

??? 🙄 ???

Qual é a área da quadra?

R: ??? 🙄 ???

Endereço => 0xB5

char cep[] = "abc" float area = 50.0

Situação

qry.c

Endereço => 0xB5

??? 🙄 ???

quadra.c

Endereço => 0xB5

char cep[] = "abc" float area = 50.0

Qual é a área da quadra?

Situação

qry.c

Endereço => 0xB5

??? 🙄 ???

quadra.c

Endereço => 0xB5

char cep[] = "abc" float area = 50.0

Qual é a área da quadra?

R: float = 50.0

Solução

- Criar uma função no arquivo original que **pega (get) e retorna o valor.**

Solução

qry.c

quadra.c

Endereço => 0xB5

??? 🙄 ???

void* quadraAtual;

float getArea(void* quadra);

Solução

qry.c

quadra.c

Endereço => 0xB5

??? 🙄 ???

```
void* quadraAtual;
```

```
float area = getArea(quadraAtual);
```

```
float getArea(void* quadra);
```

Solução

qry.c

Endereço => 0xB5

??? 🙄 ???

```
void* quadraAtual;
```

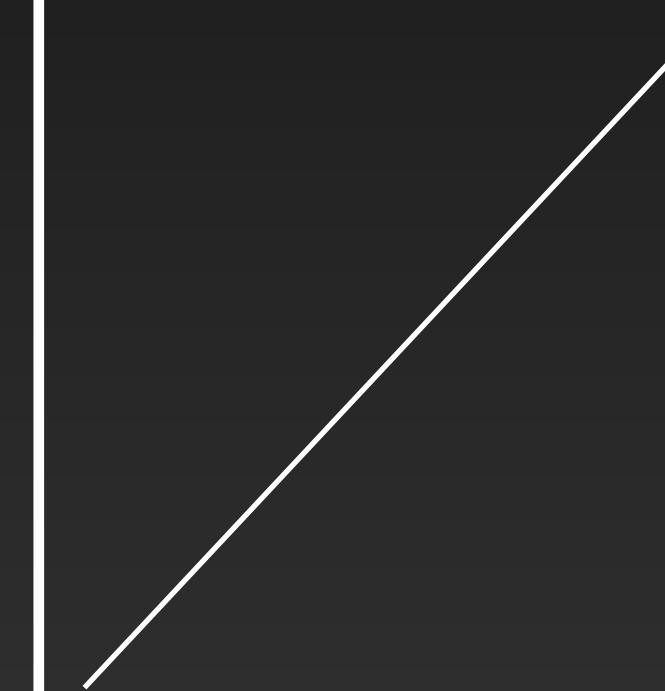
```
float area = getArea(quadraAtual);
```

quadra.c

Endereço => 0xB5

??? 🙄 ???

```
float getArea(void* quadra);
```



Solução

qry.c

Endereço => 0xB5

??? 🙄 ???

```
void* quadraAtual;
```

```
float area = getArea(quadraAtual);
```

quadra.c

Endereço => 0xB5

??? 🙄 ???

```
float getArea(void* quadra);
```

>>CASTING<<



Solução

qry.c

Endereço => 0xB5

??? 🙄 ???

```
void* quadraAtual;
```

```
float area = getArea(quadraAtual);
```

quadra.c

```
float getArea(void* quadra);
```

Endereço => 0xB5

char cep[] = "abc" float area = 50.0

Solução

qry.c

Endereço => 0xB5

??? 🙄 ???

```
void* quadraAtual;
```

```
float area = getArea(quadraAtual);
```

quadra.c

```
float getArea(void* quadra);
```

float area = 50.0

Solução

qry.c

quadra.c

Endereço => 0xB5

??? 🙄 ???

```
void* quadraAtual;
```

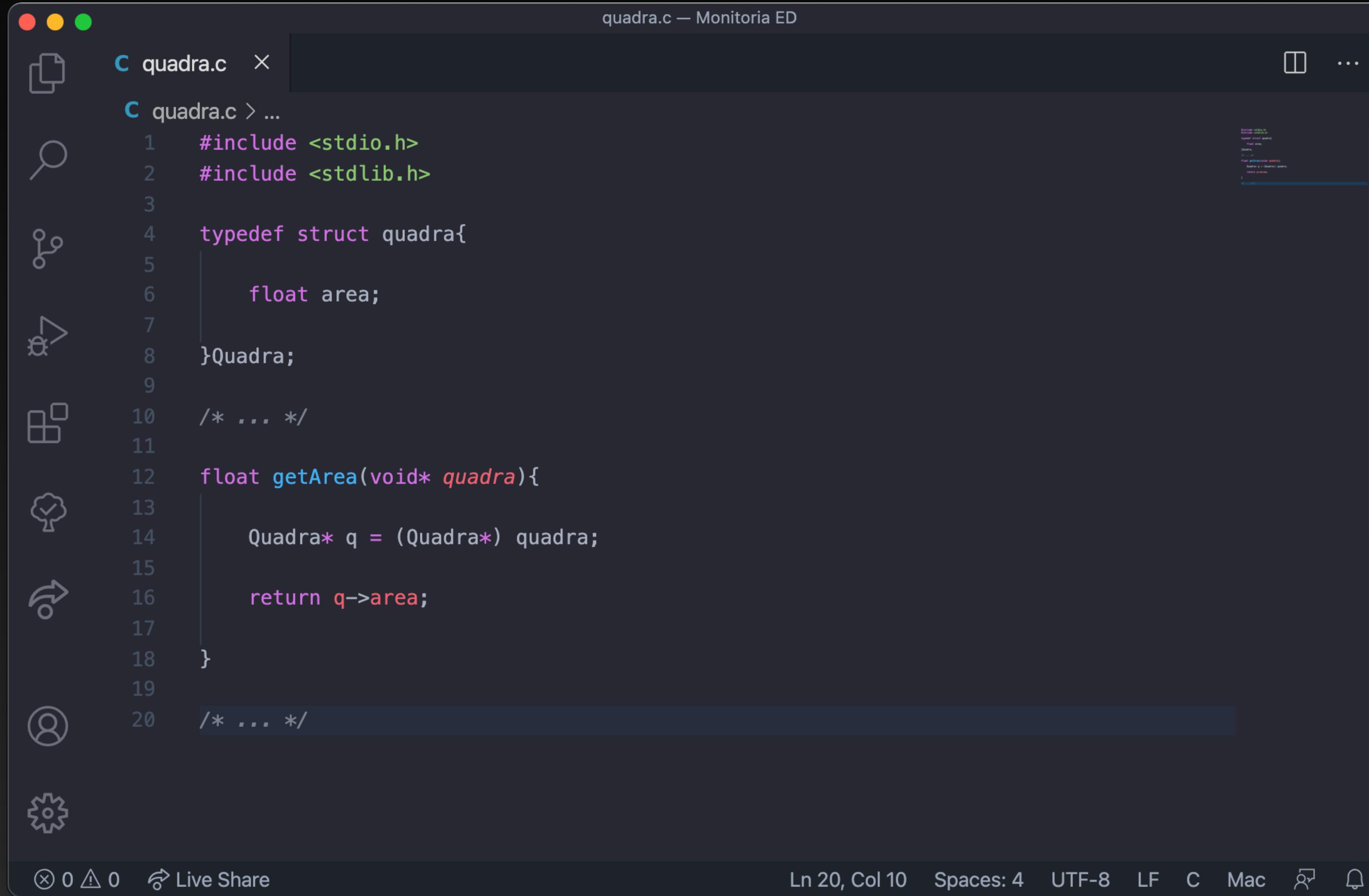
```
float area = getArea(quadraAtual);
```

```
float getArea(void* quadra);
```

float area = 50.0

← Retorna

Exemplo de Implementação (getter)



The screenshot shows a dark-themed code editor window titled "quadra.c — Monitoria ED". The code implements a struct named "quadra" with a float member "area" and a function "getArea" that returns the value of "area". The code is as follows:

```
#include <stdio.h>
#include <stdlib.h>

typedef struct quadra{
    float area;
}Quadra;

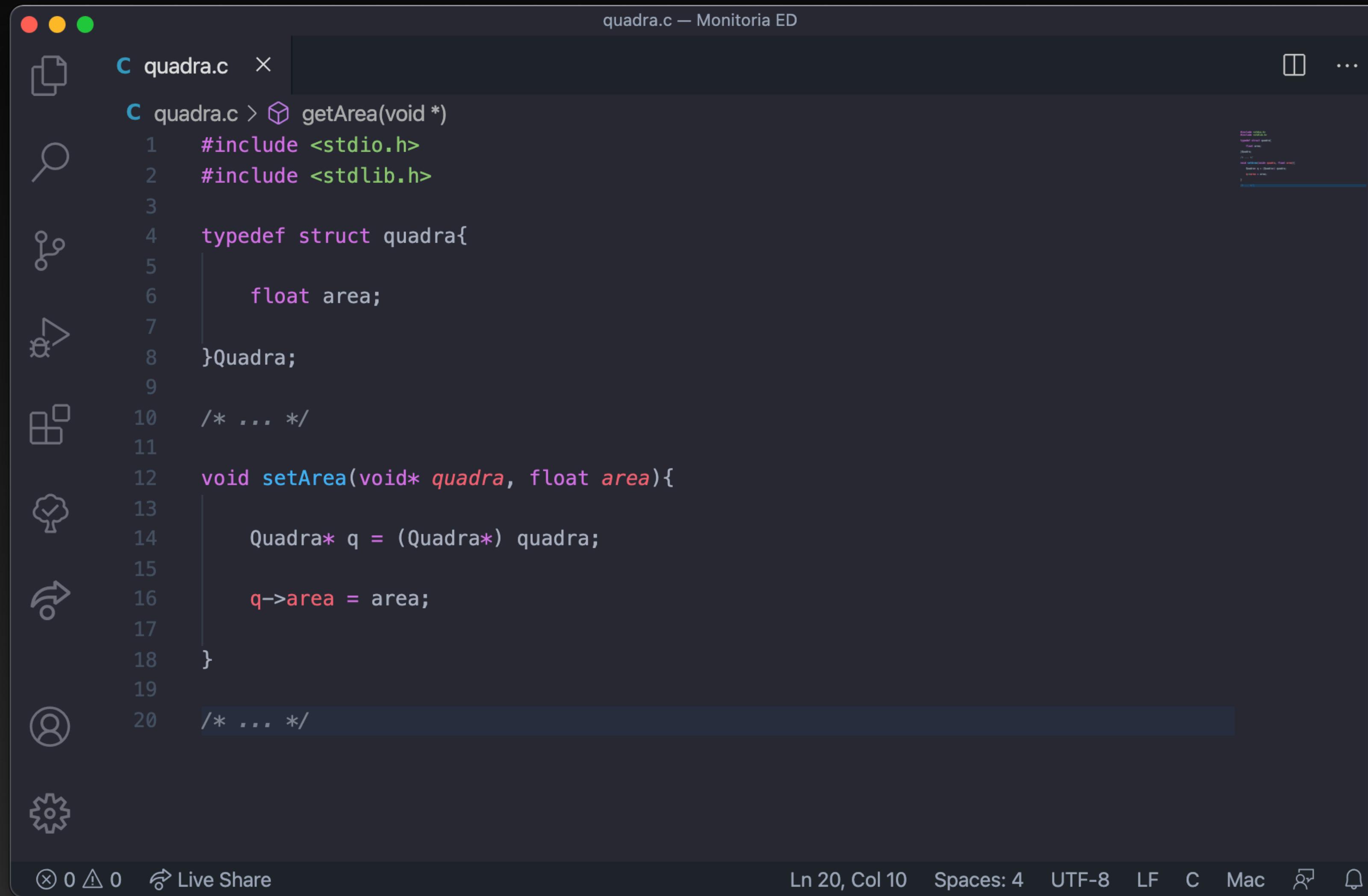
/* ... */

float getArea(void* quadra){
    Quadra* q = (Quadra*) quadra;
    return q->area;
}

/* ... */
```

The status bar at the bottom indicates the file is 20 lines long, column 10, with spaces: 4, encoding UTF-8, line endings LF, and character counts C and Mac.

Exemplo de Implementação (setter)



The screenshot shows a dark-themed code editor window titled "quadra.c — Monitoria ED". The code implements a structure named "quadra" and a setter function "setArea".

```
quadra.c — Monitoria ED

quadra.c  X
quadra.c > ⚡ getArea(void *)
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 typedef struct quadra{
5     float area;
6 }Quadra;
7
8 /* ... */
9
10 void setArea(void* quadra, float area){
11     Quadra* q = (Quadra*) quadra;
12     q->area = area;
13 }
14
15 /* ... */
16
17
18
19
20 /* ... */
```

The code includes standard library headers, defines a structure "quadra" with a single float member "area", and provides a "setArea" function that takes a pointer to a "quadra" structure and a float value, then assigns the value to the "area" member.

Getters and Setters

- Agora fica fácil manipular dados entre arquivos 😊

Typedef

Typedef

- Você já viu que trabalhar com ponteiro void é fácil.
- Mas as coisas podem ficar um pouco desorganizadas...

Typedef

```
void calcularDistancia(void* figura1, void* figura2, void* lista, void* lista2);
```

Typedef

```
void calcularDistancia(void* figura1, void* figura2, void* lista, void* lista2);
```

Como saber a qual tipo pertence cada varável?



Typedef

Solução 1

```
void calcularDistancia(void* circulo, void* retangulo, void* listaC, void* listaR);
```

Renomear para variáveis com nomes significativos

Typedef

Solução 2

Usar `typedef`

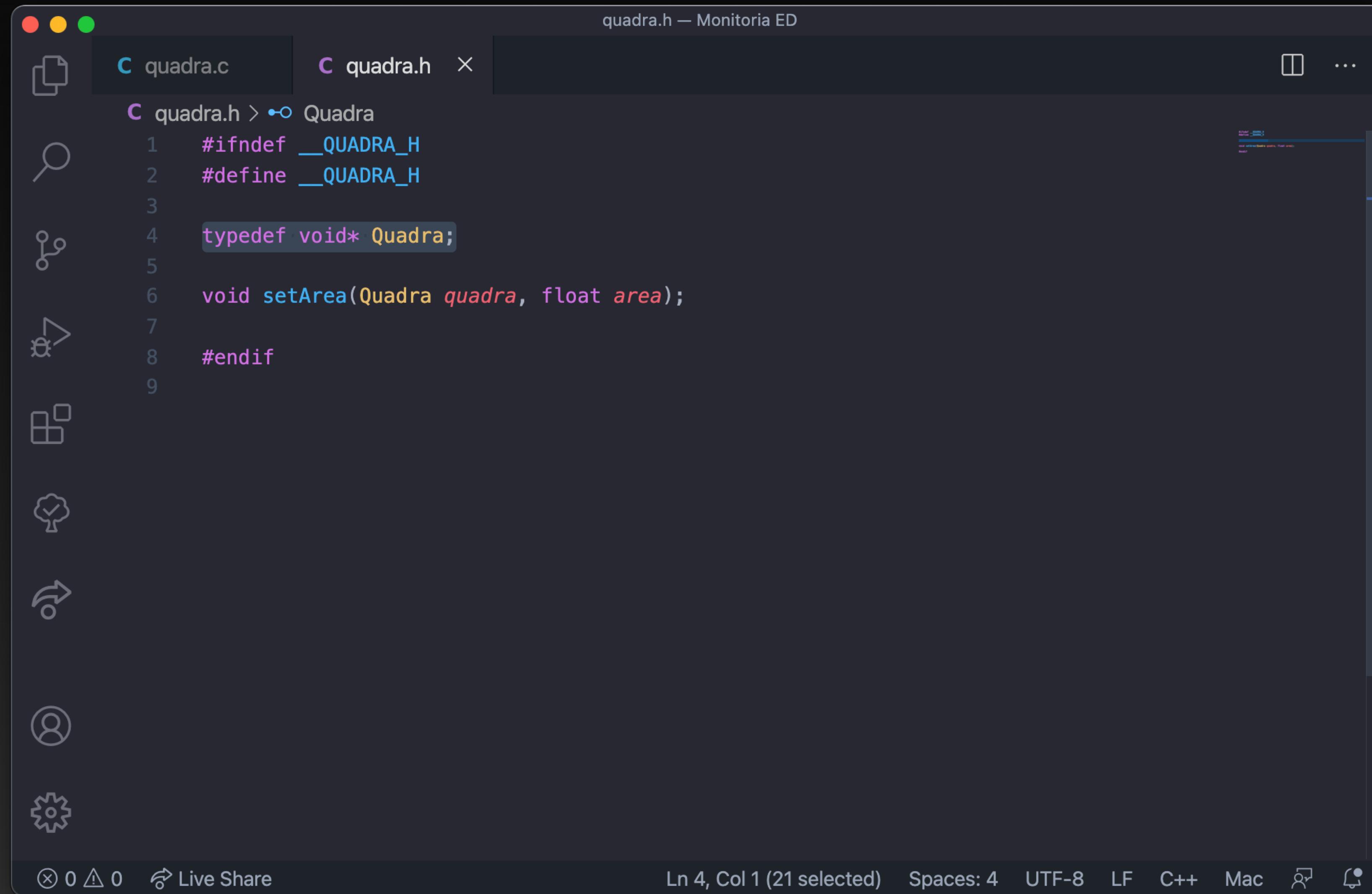
Typedef

- O typedef é capaz de nomear tipos a **qualquer** tipo de variável - não apenas structs!

Typedef

- O typedef é capaz de nomear tipos a **qualquer** tipo de variável - não apenas structs!
- Para **melhorar a organização** do código, vamos **nomear** nossos ponteiros void.

Typedef



The screenshot shows a dark-themed code editor window titled "quadra.h — Monitoria ED". The left sidebar contains icons for file operations like new, open, save, and search. The main pane displays the following C code:

```
#ifndef __QUADRA_H
#define __QUADRA_H

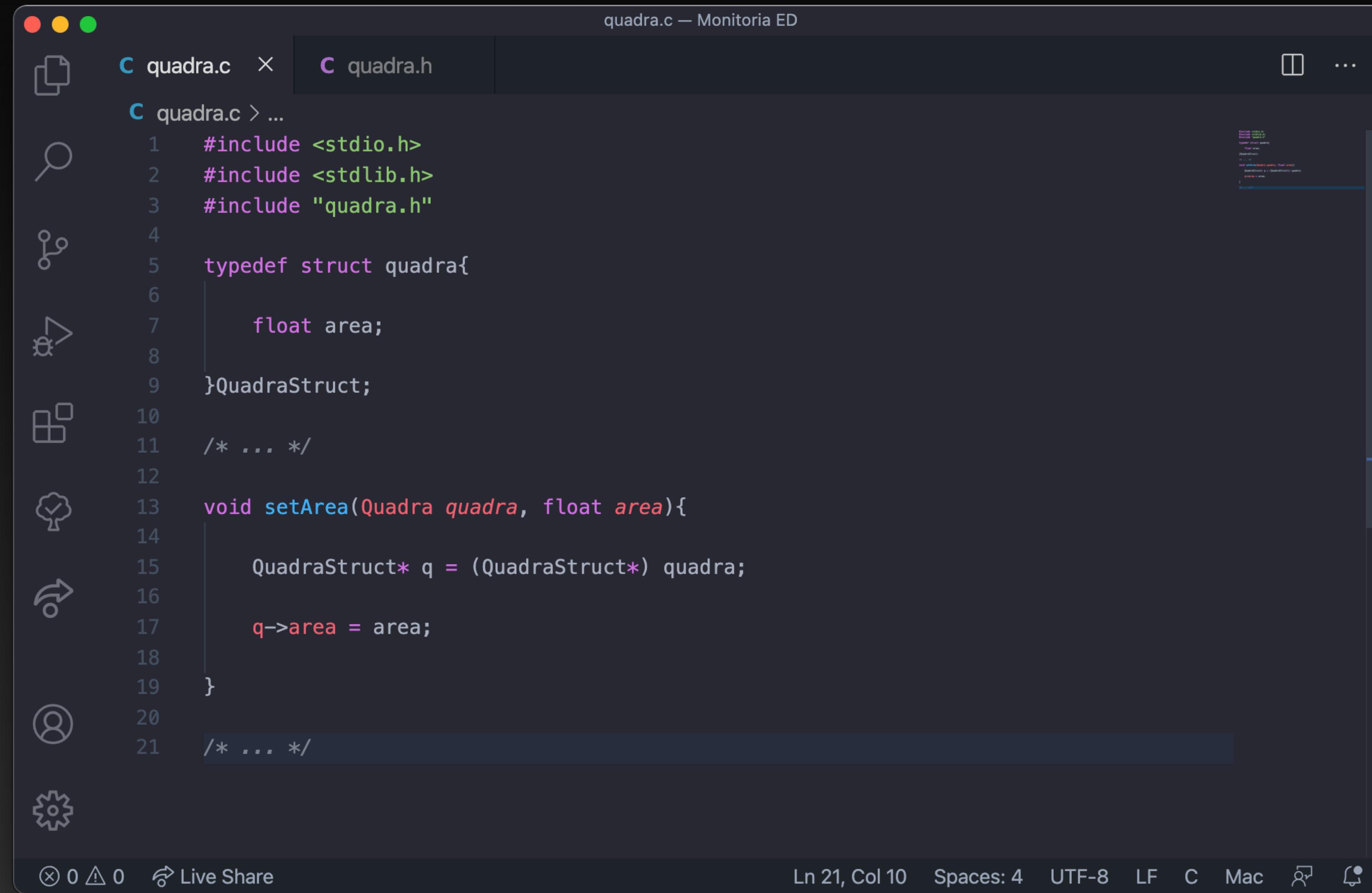
typedef void* Quadra;

void setArea(Quadra quadra, float area);

#endif
```

The code editor interface includes status bars at the bottom showing line and column counts (Ln 4, Col 1), character counts (Spaces: 4), and encoding (UTF-8). It also shows file type indicators (LF, C++, Mac) and other standard editor controls.

Typedef



The screenshot shows a dark-themed code editor window titled "quadra.c — Monitoria ED". The file "quadra.c" is open, displaying the following C code:

```
quadra.c > ...
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include "quadra.h"
4
5 typedef struct quadra{
6
7     float area;
8
9 }QuadraStruct;
10
11 /* ... */
12
13 void setArea(Quadra quadra, float area){
14
15     QuadraStruct* q = (QuadraStruct*) quadra;
16
17     q->area = area;
18
19 }
20
21 /* ... */
```

The code defines a struct named "quadra" with a single float member "area". It then declares a function "setArea" that takes a "quadra" object and a float, and sets the "area" member of the struct to the given value.

TypeDef

⚠ AVISO ⚠

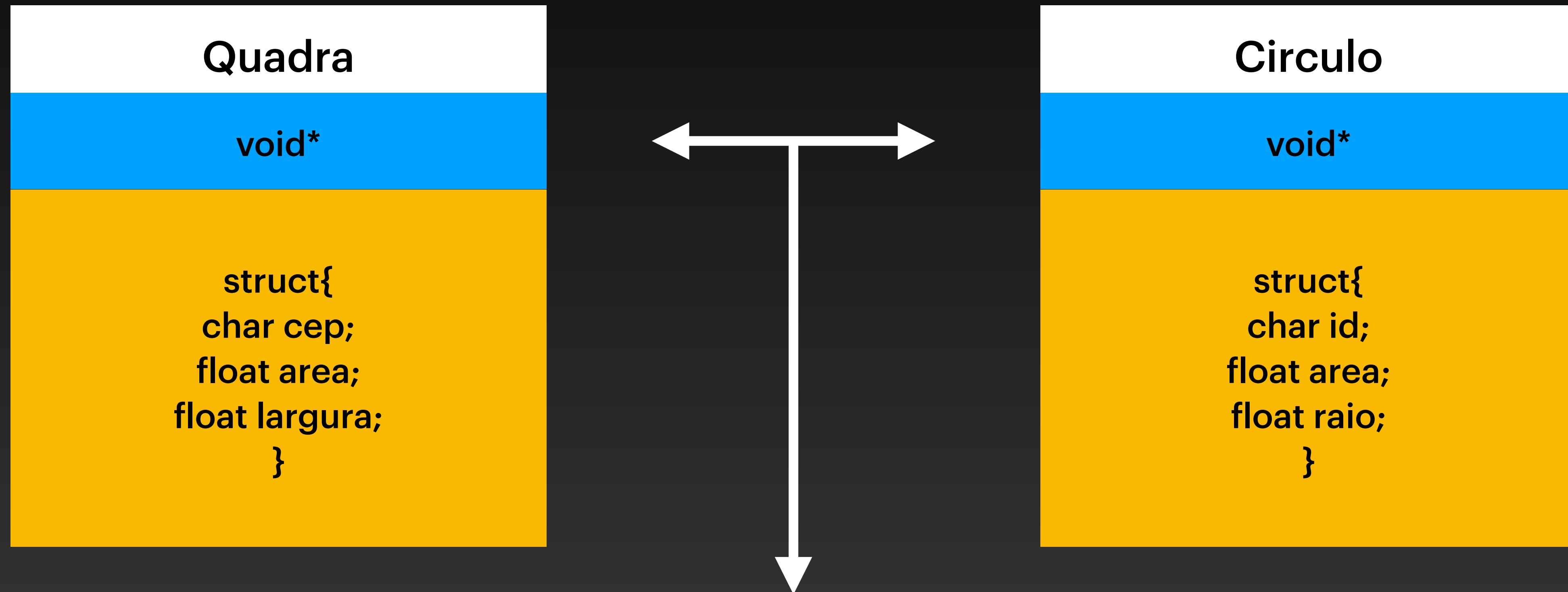
- O `typedef` **não garante** verificação de tipo pelo nome atribuido.
- Ou seja, caso exista:

```
typedef void* Quadra;  
typedef void* Hidrante;
```

- Os dois são o mesmo tipo (**void***) mesmo possuindo nomes diferentes!

Typedef

⚠ AVISO ⚠



Tipos primitivos iguais = Variáveis iguais

Typedef

⚠ AVISO ⚠

- Principal cuidado: evitar com que o conteúdo seja sobrescrito.

Typedef

- Parabéns! Agora você já sabe como nomear os tipos de suas variáveis ✨

Implementação

Código disponível no Git

<https://github.com/FernandoMPN/MonitoriaED>

You have mastered **Typedef Void***
Parabéns!

