Algoritmos
e
Estrutura de Dados

Profª. Angela Abreu Rosa de Sá, Drª.

Contato: angelaabreu@gmail.com

Lembrar ...

É um processo INDIVIDUAL!

Depende apenas de você!

IMPORTANTE

- Não se compare com o colega;
- Pergunte QUALQUER dúvida;
- Implemente os exercícios propostos;
- Pratique sempre...inclusive em casa!
- Não desista!

Ementa

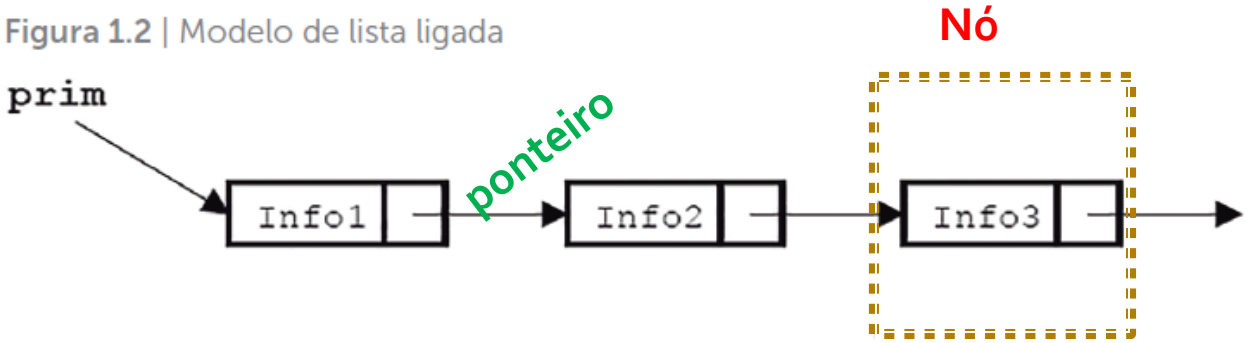

1
Listas Ligadas
Definição
Operações
Listas duplamente ligadas
2
Pilhas e filas
Definição
Operações com pilhas
Operações com filas
3
Tabelas de espalhamento
Definição
Operações
Otimização
4
Armazenamento associativo
Definição
Mapas com lista
Mapas com espalhamento

Algoritmos e Estrutura de Dados

Sumário

Listas Ligadas (Encadeadas)

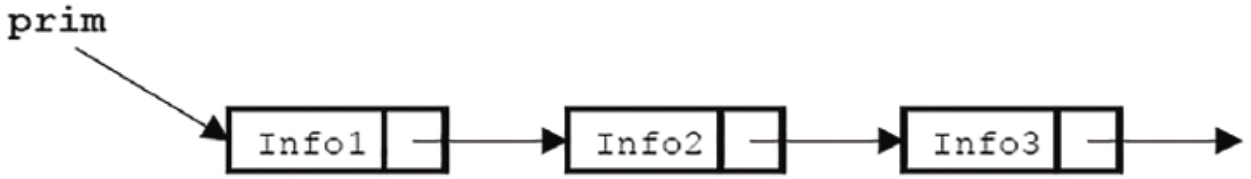
Figura 1.2 | Modelo de lista ligada



E composta de um conjunto de dados dispostos por uma **sequencia de nós**, em que a relação de sucessão desses elementos e determinada por um **ponteiro** que indica a posição do próximo elemento

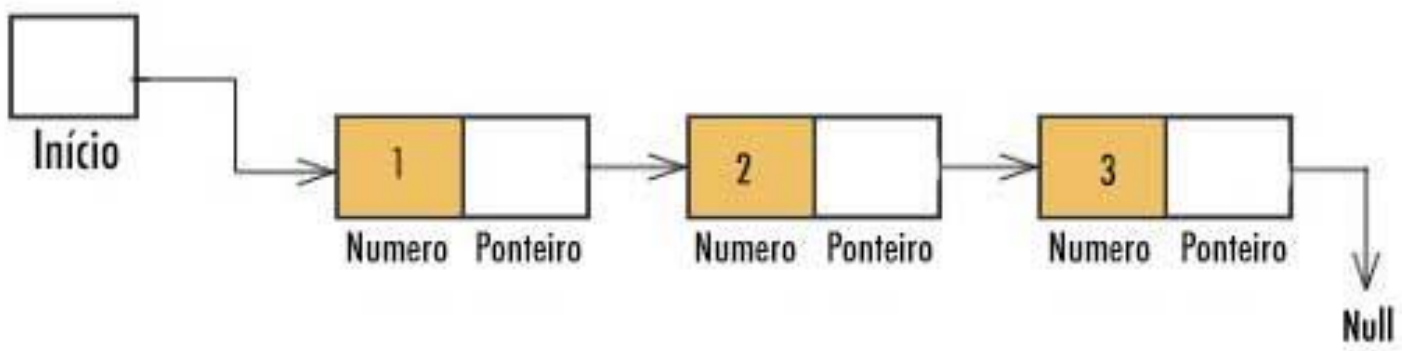
Listas Ligadas (Encadeadas)

Toda lista precisa ter sua **estrutura definida**, sabendo que cada nó é composto por um conjunto de informações de tipos diferentes e outro de valor inteiro para o ponteiro.




```
struct lista {
int info;
struct lista* prox;
};
```

Listas Ligadas (Encadeadas)



Listas Ligadas (Encadeadas)

Exemplificando

Exemplo de declaração para criar uma lista em C:

```
/*Cria a estrutura da lista*/
struct alunos {
char nome[25];
struct alunos* prox;
};
```

Será criada uma struct (registro) **alunos**:

- Na struct, temos a variável nome do tipo char, que será nossa informação;

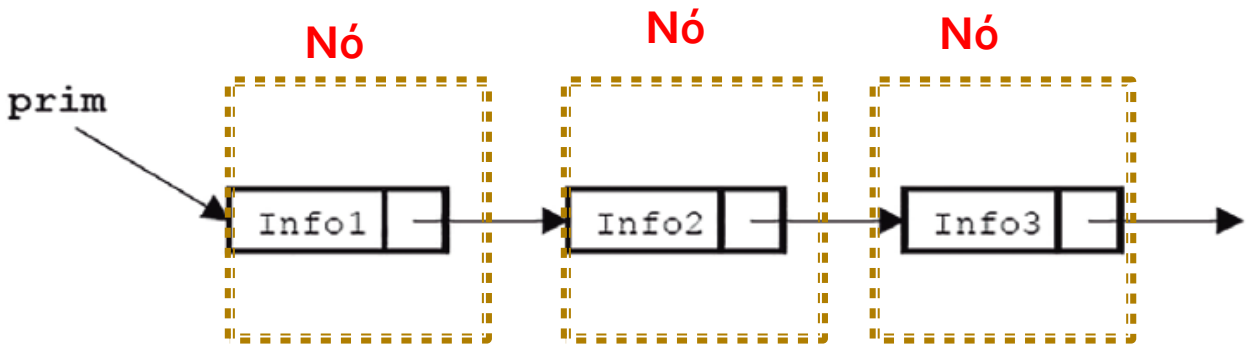
- Temos outra struct prox com ponteiro para a própria struct alunos, para receber o endereço de apontamento da próxima informação.

Listas Ligadas (Encadeadas)

Figura 1.4 | Exemplo de programa em C com *malloc( )* e *sizeof*

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

int main() {

    int *p;
    p=(int *) malloc(sizeof(int));

    if (!p) {
        printf("Erro de memoria insuficiente");
    }else{
        printf("Memoria alocada com sucesso");
    }

    return 0;
}
```

Passo 1: criar a estrutura dos nós

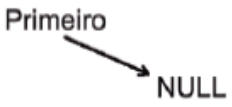



```
struct noLista
{
   int informacao;
   struct noLista *proximo;
};
```

Passo 2: criar o ponteiro para o primeiro nó da lista

Quando uma lista esta sem nós, e definida como vazia ou nula, é considerado ponteiro nulo.

Figura 1.3 | Lista vazia com ponteiro nulo




```c
#include <stdio.h>

struct noLista
{
  int informacao;
  struct noLista *proximo;
};

int main()
{
    struct noLista *PrimeiroNoDaLista = NULL;
}
```

Inserir novos elementos (nós) na Lista

- Inserir um novo elemento no início da Lista
- Inserir um novo elemento em uma determinada posição da Lista
- Inserir um novo elemento no final da Lista

Passo 3: Inserir no INÍCIO da Lista

```c
struct noLista * inserirInicioLista (struct noLista *InicioDaLista, int NovoNumero)
 {
    //alocar memória para o novo Nó da Lista
    struct noLista* novoNo = (struct noLista*) malloc(sizeof(struct noLista));

    //Inserir as informação para o novo nó
    novoNo->informacao = NovoNumero;
    //Apontar o campo "próximo" do novo nó para o local que o InicioDaLista apontava
    novoNo->proximoNo = InicioDaLista;

    return novoNo;
}



int main()
{
    struct noLista *PrimeiroNoDaLista = NULL;

    //inserir a informaçao "10" na lista
   //---- o Primeiro da lista irá apontar agora para o novo nó que foi criado
    PrimeiroNoDaLista = inserirInicioLista (PrimeiroNoDaLista, 10);

}
```

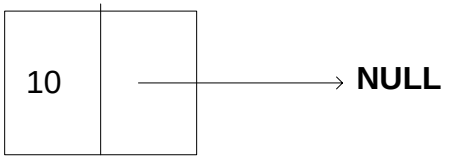
Passo 3: Inserir no INÍCIO da Lista

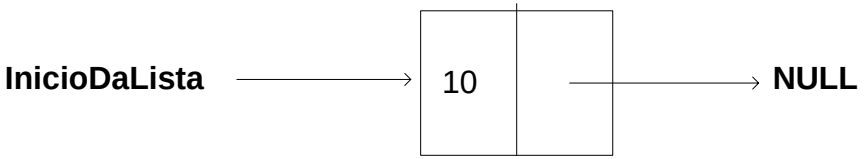
```c
int main()
{
    struct noLista *PrimeiroNoDaLista = NULL;

    //inserir a informaçao "10" na lista
   //---- o Primeiro da lista irá apontar agora para o novo nó que foi criado
    PrimeiroNoDaLista = inserirInicioLista (PrimeiroNoDaLista, 10);

}
```

InicioDaLista = NULL;

novoNo

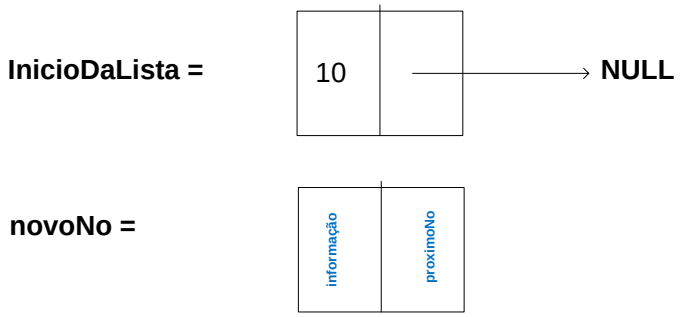
InicioDaLista = novoNo

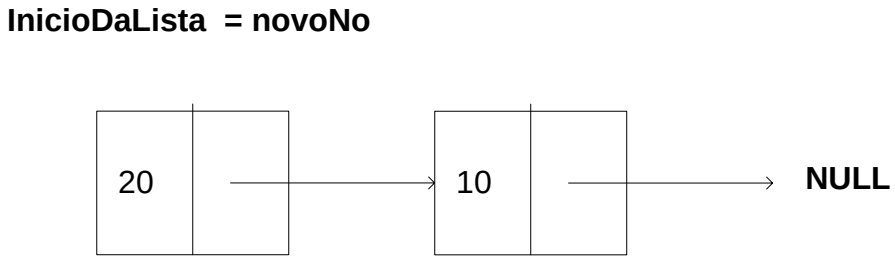


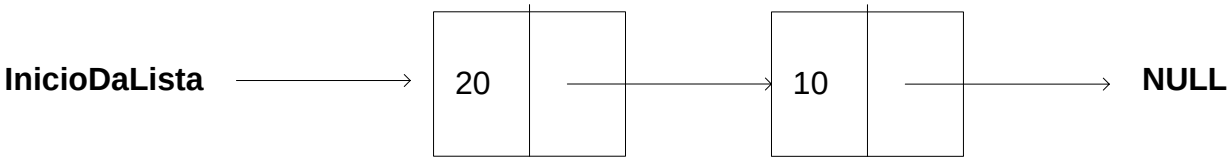


Passo 3: Inserir no INÍCIO da Lista

```c
int main()
{
    struct noLista *PrimeiroNoDaLista = NULL;

    //inserir a informaçao "10" na lista
   //---- o Primeiro da lista irá apontar agora para o novo nó que foi criado
    PrimeiroNoDaLista = inserirInicioLista (PrimeiroNoDaLista, 10);

    //inserir a informaçao "20" na lista
    //---- o Primeiro da lista irá apontar agora para o novo nó que foi criado
    PrimeiroNoDaLista= inserirInicioLista (PrimeiroNoDaLista, 20);

}

    ImprimirLista(PrimeiroNoDaLista);
```







Imprimir a Lista

```c
void ImprimirLista(struct noLista  *InicioDaLista)
{
    noLista *NoAtual = InicioDaLista; //copiar o endereço do primeiro nó da lista

    while (NoAtual != NULL) //percorrer a lista até encontrar o último nó = NULL
    {
        printf("%d -> ",NoAtual->informacao); //mostrar a informação do nó
        NoAtual = NoAtual->proximoNo; //apotar para o próximo nó da lista
    }

    printf("NULL");

}
```

```c
int main()
{
    struct noLista *PrimeiroNoDaLista = NULL;

    //inserir a informaçao "10" na lista
   //---- o Primeiro da lista irá apontar agora para o novo nó que foi criado
    PrimeiroNoDaLista = inserirInicioLista (PrimeiroNoDaLista, 10);

    //inserir a informaçao "20" na lista
    //---- o Primeiro da lista irá apontar agora para o novo nó que foi criado
    PrimeiroNoDaLista= inserirInicioLista (PrimeiroNoDaLista, 20);

   //imprimir o conteúdo da lista
   ImprimirLista(PrimeiroNoDaLista);

}
```

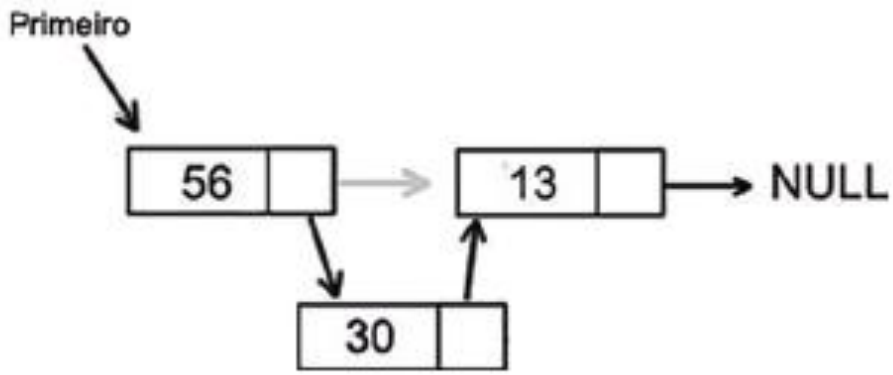
```
20 -> 10 -> NULL
--------------------------------
```

Inserir novos elementos (nós) na Lista

- Inserir um novo elemento no início da Lista
- Inserir um novo elemento em uma determinada posição da Lista
- Inserir um novo elemento no final da Lista

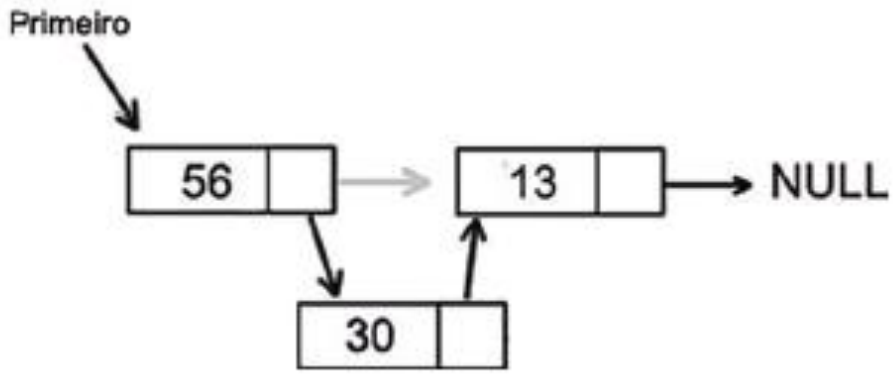
Passo 4: Inserir na posição X da Lista

```c
struct noLista * inserirPosicaoLista (struct noLista *InicioDaLista, int NovoNumero,int PosicaoInsercao)
{
    int contador = 0; //auxiliar para contar as posições de inserção
    struct noLista *PercorreLista = InicioDaLista; //cópia do endereço do início da lista
    struct noLista* novoNo = (struct noLista*)malloc(sizeof(struct noLista)); //alocar memória para o novoNo

    //percorrer a lista enquanto não encontrar a posição de inserção
    //ou enquanto não alcançar o último elemento da lista
    while ((contador < PosicaoInsercao) && (PercorreLista->proximoNo != NULL))
    {
        PercorreLista = PercorreLista->proximoNo; //andar com o ponteiro para o próximo nó
        contador++; //atualizar o contador
    }

    //Inserir a nova informação no novo nó
    novoNo->informacao = NovoNumero;
    //o novo nó deve apontar para o próximo nó do Percorre Lista
    novoNo->proximoNo  = PercorreLista->proximoNo;
    //Percorre Lista deve apontar para o novo nó
    PercorreLista->proximoNo = novoNo;

    return InicioDaLista;
}
```



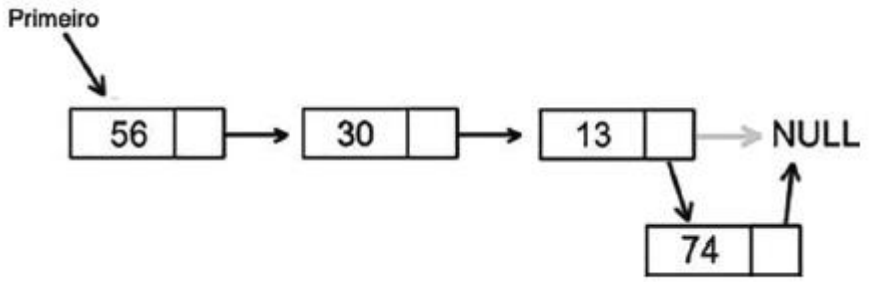
Passo 4: Inserir na posição X da Lista

```
int main()
{
    struct noLista *PrimeiroNoDaLista = NULL;

    //inserir a informaçao "10" na lista
   //---- o Primeiro da lista irá apontar agora para o novo nó que foi criado
    PrimeiroNoDaLista = inserirInicioLista (PrimeiroNoDaLista, 13);

    //inserir a informaçao "20" na lista
    //---- o Primeiro da lista irá apontar agora para o novo nó que foi criado
    PrimeiroNoDaLista= inserirInicioLista (PrimeiroNoDaLista, 56);

    PrimeiroNoDaLista= inserirPosicaoLista (PrimeiroNoDaLista, 30,0) ;

   //imprimir o conteúdo da lista
   ImprimirLista(PrimeiroNoDaLista);

}
```



Passo 5: Inserir no FINAL da Lista

```
struct noLista* inserirFimLista(struct noLista* InicioLista, int NovoNumero)
{
    struct noLista *PercorreLista = InicioLista; //cópia do início da lista
    struct noLista* novoNo = (struct noLista*)malloc(sizeof(struct noLista)); //
    while (PercorreLista->proximoNo != NULL) //percorrer a lista até encontrar o último nó
    {
      PercorreLista = PercorreLista->proximoNo;

    }
    novoNo->informacao = NovoNumero; //inserir a nova informação
    novoNo->proximoNo = PercorreLista->proximoNo; //apontar para o próximo nó que o último estava apontando (NULL)
    PercorreLista->proximoNo = novoNo; // o que era o último nó (agora será o penúltimo!) aponta para o novo nó

    return InicioLista;
}
```

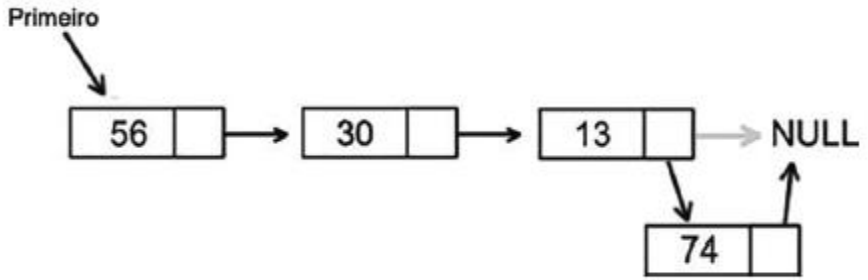


Elaborada pelo autor

Passo 5: Inserir no FINAL da Lista

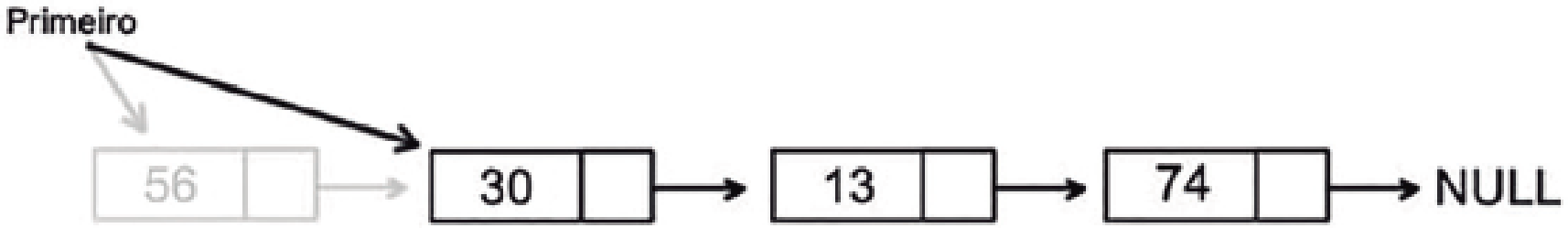
```
int main()
{
    struct noLista *PrimeiroNoDaLista = NULL;

    //inserir a informaçao "10" na lista
   //---- o Primeiro da lista irá apontar agora para o novo nó que foi criado
    PrimeiroNoDaLista = inserirInicioLista (PrimeiroNoDaLista, 13);

    //inserir a informaçao "20" na lista
    //---- o Primeiro da lista irá apontar agora para o novo nó que foi criado
    PrimeiroNoDaLista= inserirInicioLista (PrimeiroNoDaLista, 56);

    //inserir o número 30 após a posição 0
    PrimeiroNoDaLista= inserirPosicaoLista (PrimeiroNoDaLista, 30,0) ;

    //inserir o número 74 no fim da lista
    PrimeiroNoDaLista =  inserirFimLista(PrimeiroNoDaLista, 74);

   //imprimir o conteúdo da lista
   ImprimirLista(PrimeiroNoDaLista);

}
```

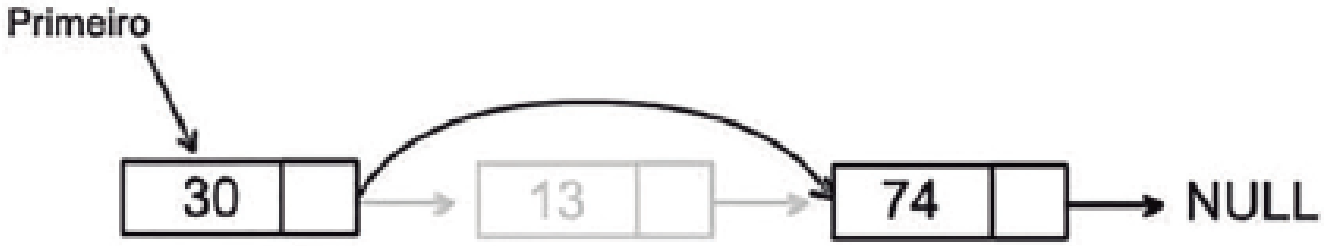
```
56 -> 30 -> 13 -> 74 -> NULL
--------------------------------
```



elaborada pelo autor

Passo 6: REMOVER elementos da lista

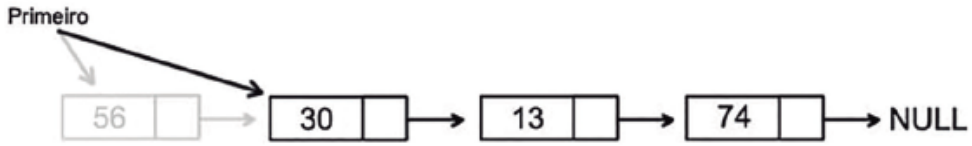




Passo 6: REMOVER elementos da lista

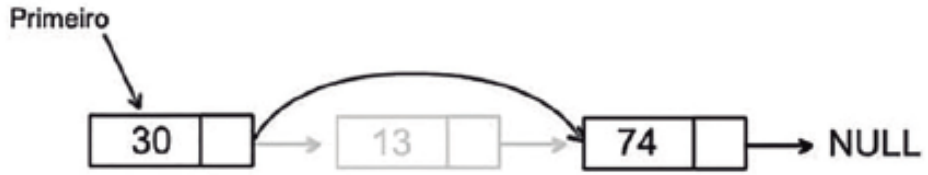
```c
struct noLista* remove (struct noLista* InicioLista, int ElementoParaRemover)
{
    struct noLista* NoAnterior = NULL; //ponteiro para o nó anterior
    struct noLista* PercorreLista = InicioLista; //cópia do início da lista

    //procurar o elementou ou até chegar no final da lista
    while (PercorreLista != NULL && PercorreLista->informacao != ElementoParaRemover)
    {
        NoAnterior = PercorreLista; //manter uma cópia do endereço do nó anterior
        PercorreLista = PercorreLista->proximoNo; //percorrer a lista
    }

    if (PercorreLista == NULL ) //não encontrou o número na lista para ser removido
        return InicioLista;

    if (NoAnterior == NULL) //significa que o elemento procurado é  o primeiro da lista
    {
        InicioLista = PercorreLista->proximoNo; //atualizar o início da lista
    }
    else {
            NoAnterior->proximoNo = PercorreLista->proximoNo; //"ligar" o próximo nó do anterior com o próximo do elemento que será removido
        }

    free(PercorreLista); //liberar região de memória do nó removido

  return InicioLista;
}
```

```c
int main()
{
    struct noLista *PrimeiroNoDaLista = NULL;

    //inserir a informaçao "10" na lista
   //---- o Primeiro da lista irá apontar agora para o novo nó que foi criado
    PrimeiroNoDaLista = inserirInicioLista (PrimeiroNoDaLista, 13);

    //inserir a informaçao "20" na lista
    //---- o Primeiro da lista irá apontar agora para o novo nó que foi criado
    PrimeiroNoDaLista= inserirInicioLista (PrimeiroNoDaLista, 56);

    //inserir o número 30 após a posição 0
    PrimeiroNoDaLista= inserirPosicaoLista (PrimeiroNoDaLista, 30,0) ;

    //inserir o número 74 no fim da lista
    PrimeiroNoDaLista =  inserirFimLista(PrimeiroNoDaLista, 74);

    PrimeiroNoDaLista  = remove (PrimeiroNoDaLista, 56);

   //imprimir o conteúdo da lista
   ImprimirLista(PrimeiroNoDaLista);

}
```




```
C:\Users\angel\OneDrive\Desktop\E
30 -> 13 -> 74 -> NULL
--------------------------------
```

```c
  PrimeiroNoDaLista  = remove (PrimeiroNoDaLista, 74);

 //imprimir o conteúdo da lista
 ImprimirLista(PrimeiroNoDaLista);
```




```
30 -> 13 -> NULL
--------------------------------
```

Passo 7: PROCURAR elementos na lista

```c
struct noLista* ProcurarElementoLista(struct noLista* InicioLista, int ElementoProcurado)
{
    struct noLista* PercorreLista;
    PercorreLista = InicioLista;//copia do inicio da lista

    while(PercorreLista!= NULL) //enquanto existir nó na lista
    {
      if (PercorreLista->informacao == ElementoProcurado) //se encontrou o elemento procurado
        return PercorreLista; //retorna o nó encontrado

      else PercorreLista = PercorreLista->proximoNo; //continuar percorrendo a lista
    }

    //se saiu o while, é porque não encontrou o número! Retornar NULL;
    return NULL;
}
```

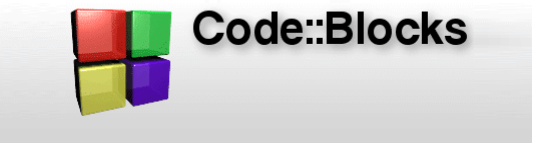
```c
//imprimir o conteúdo da lista
ImprimirLista(PrimeiroNoDaLista);


if (ProcurarElementoLista(PrimeiroNoDaLista, 30) == NULL)
 {
    printf("\n Elemento não encontrado");
}
 else
 {
   printf("\n Elemento encontrado");
 }
```

```
30 -> 13 -> NULL
 Elemento encontrado
--------------------------------
```

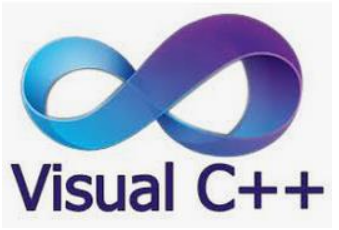
IDE – Ambiente de Desenvolvimento

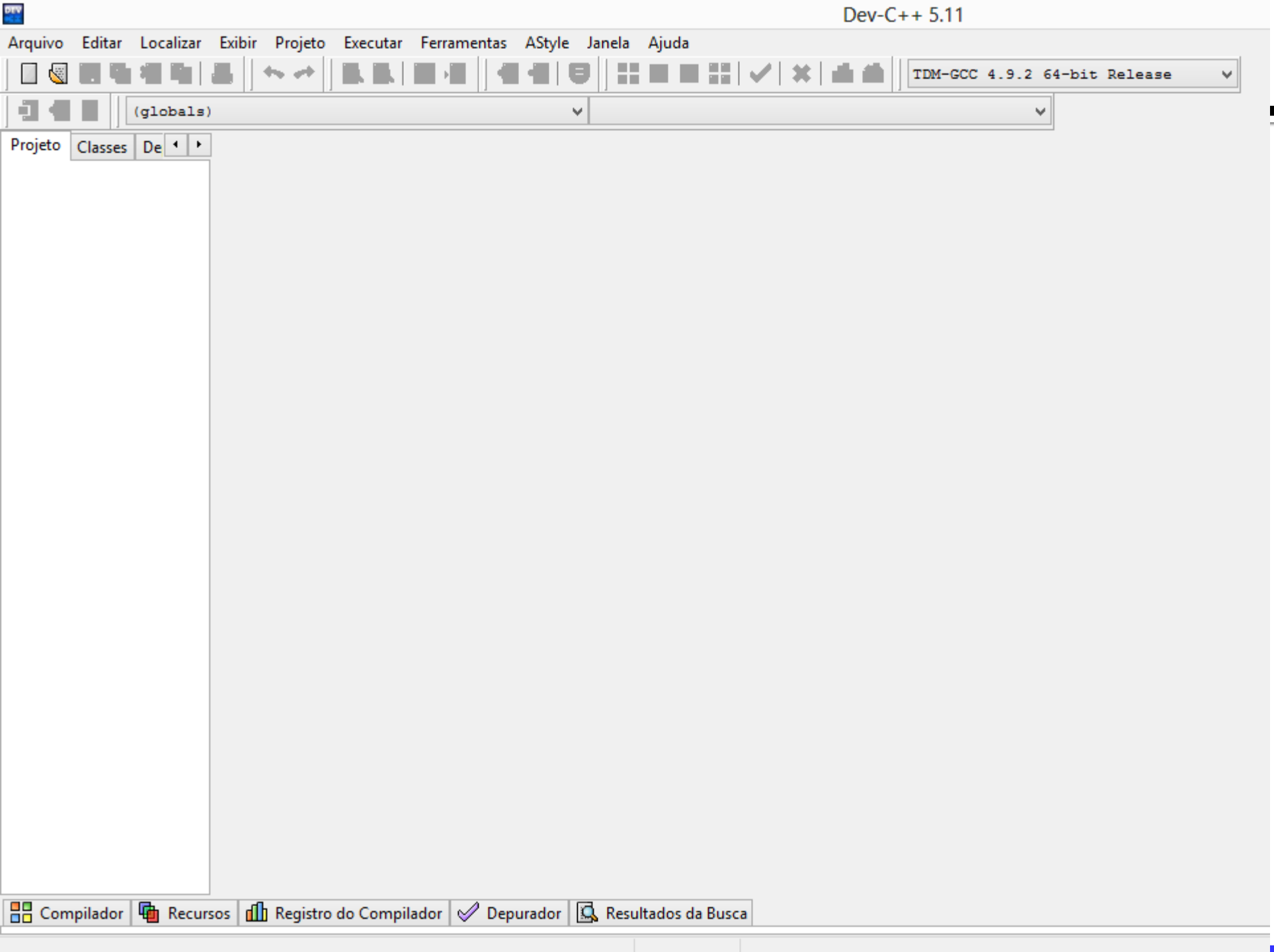
Integrated Development Environment

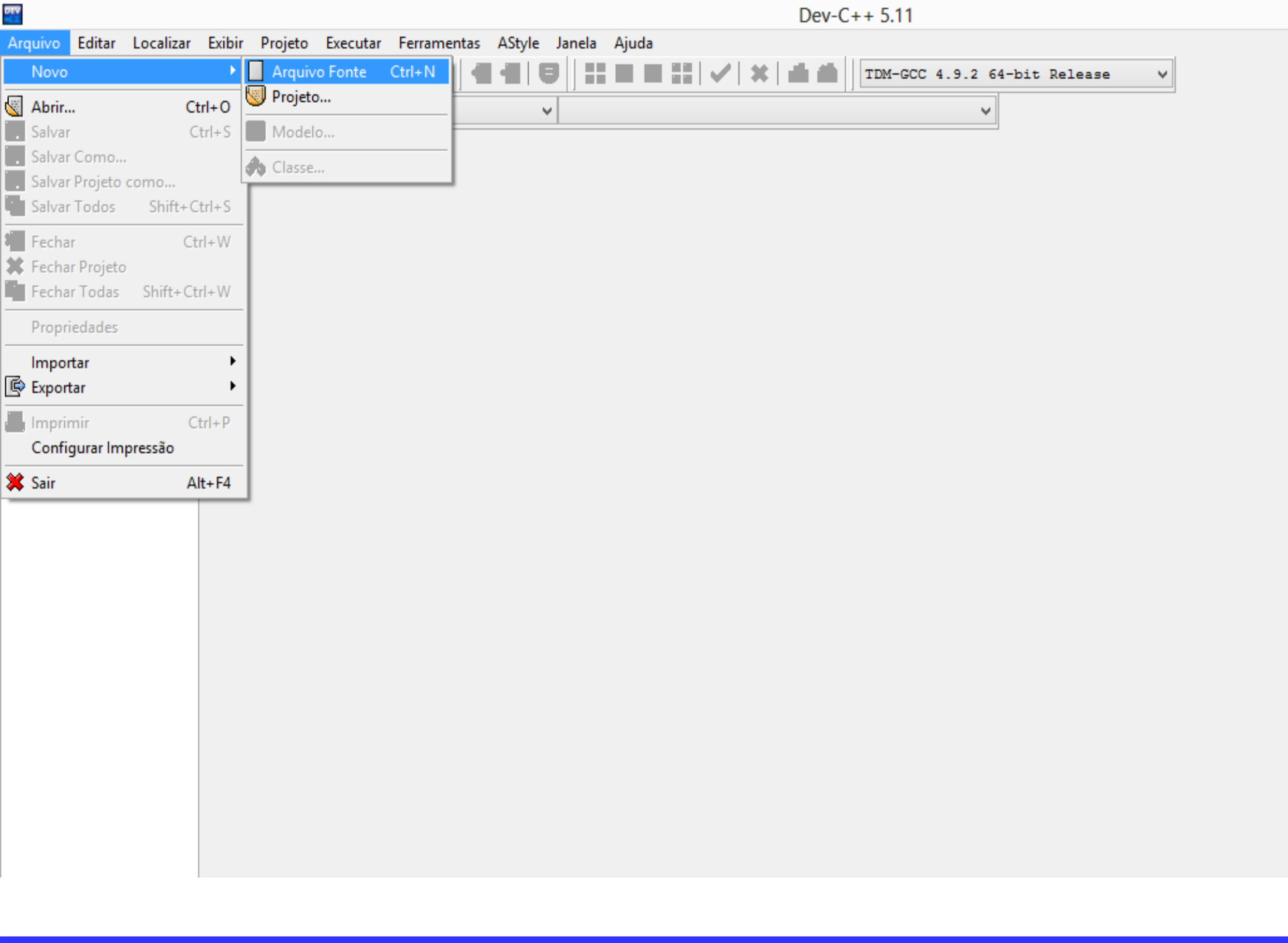
DevC++

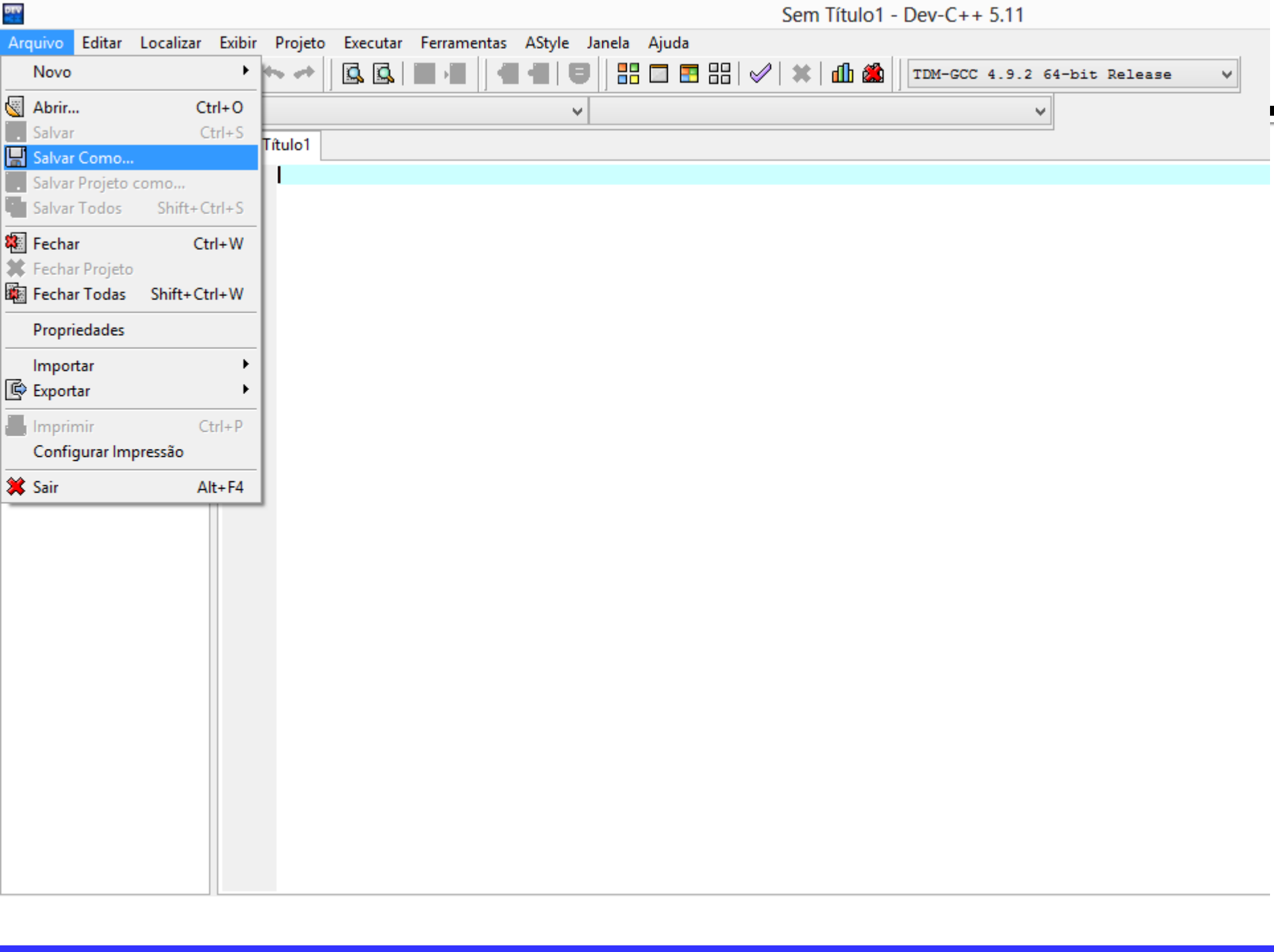


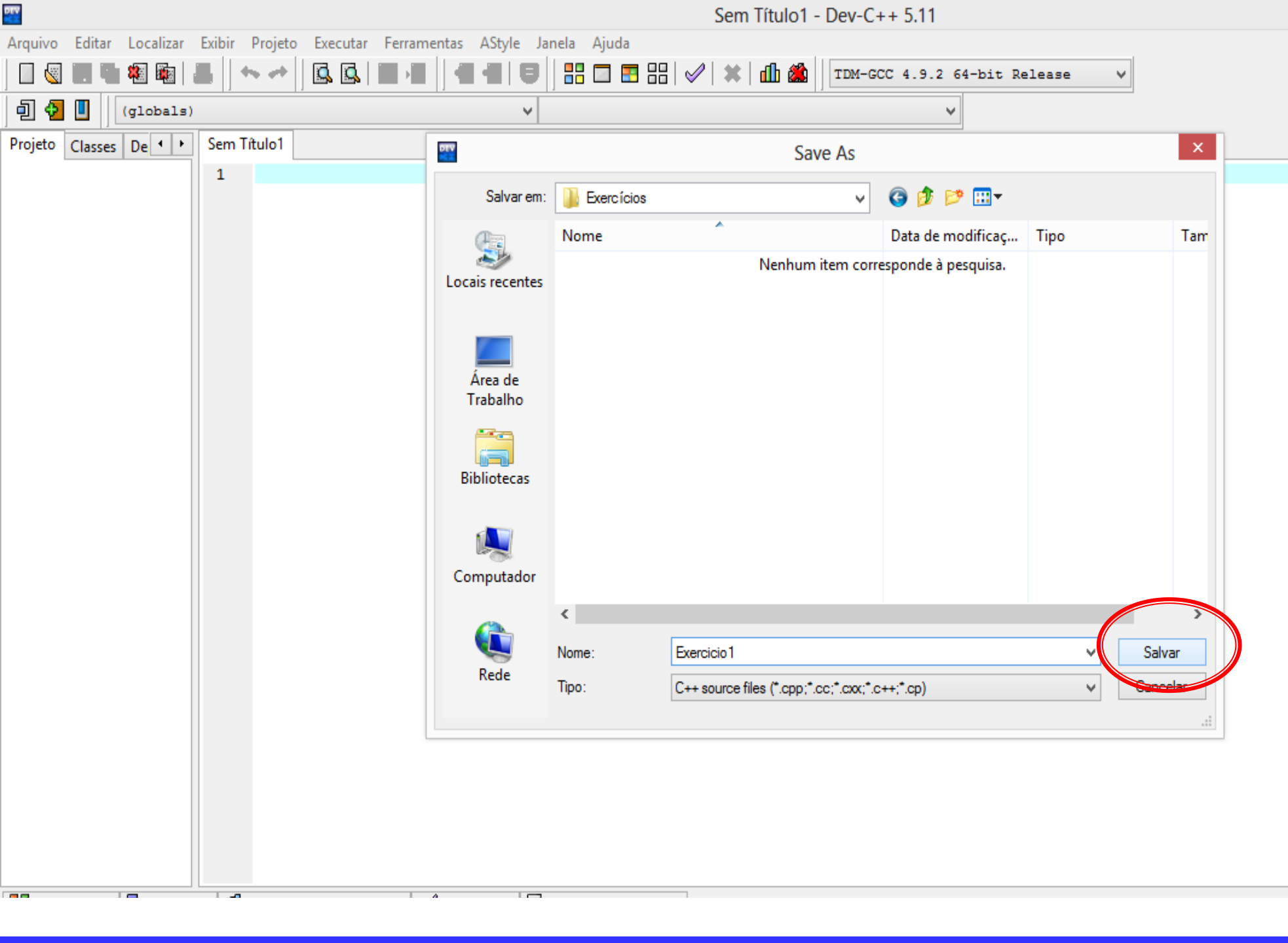
C++ Builder




Dev-C++ 5.11
Arquivo Editar Localizar Exibir Projeto Executar Ferramentas AStyle Janela Ajuda
TDM-GCC 4.9.2 64-bit Release
(globals)
Projeto Classes De
Compilador Recursos Registro do Compilador Depurador Resultados da Busca


Dev-C++ 5.11
Arquivo
Editar
Localizar
Exibir
Projeto
Executar
Ferramentas
AStyle
Janela
Ajuda
Novo
Abrir... Ctrl+O
Salvar Ctrl+S
Salvar Como...
Salvar Projeto como...
Salvar Todos Shift+Ctrl+S
Fechar Ctrl+W
Fechar Projeto
Fechar Todas Shift+Ctrl+W
Propriedades
Importar
Exportar
Imprimir Ctrl+P
Configurar Impressão
Sair Alt+F4
Arquivo Fonte Ctrl+N
Projeto...
Modelo...
Classe...
TDM-GCC 4.9.2 64-bit Release


Sem Título1 - Dev-C++ 5.11
Arquivo
Editar
Localizar
Exibir
Projeto
Executar
Ferramentas
AStyle
Janela
Ajuda
Novo
Abrir... Ctrl+O
Salvar Ctrl+S
Salvar Como...
Salvar Projeto como...
Salvar Todos Shift+Ctrl+S
Fechar Ctrl+W
Fechar Projeto
Fechar Todas Shift+Ctrl+W
Propriedades
Importar
Exportar
Imprimir Ctrl+P
Configurar Impressão
Sair Alt+F4
TDM-GCC 4.9.2 64-bit Release
Título1


Sem Título1 - Dev-C++ 5.11
Arquivo Editar Localizar Exibir Projeto Executar Ferramentas AStyle Janela Ajuda
TDM-GCC 4.9.2 64-bit Release
(globals)
Projeto Classes De
Sem Título1
1
Save As
Salvar em: Exercícios
Nome
Data de modificaç...
Tipo
Tam
Nenhum item corresponde à pesquisa.
Locais recentes
Área de Trabalho
Bibliotecas
Computador
Rede
Nome: Exercicio1
Tipo: C++ source files (*.cpp;*.cc;*.cxx;*.c++;*.cp)
Salvar
Cancelar

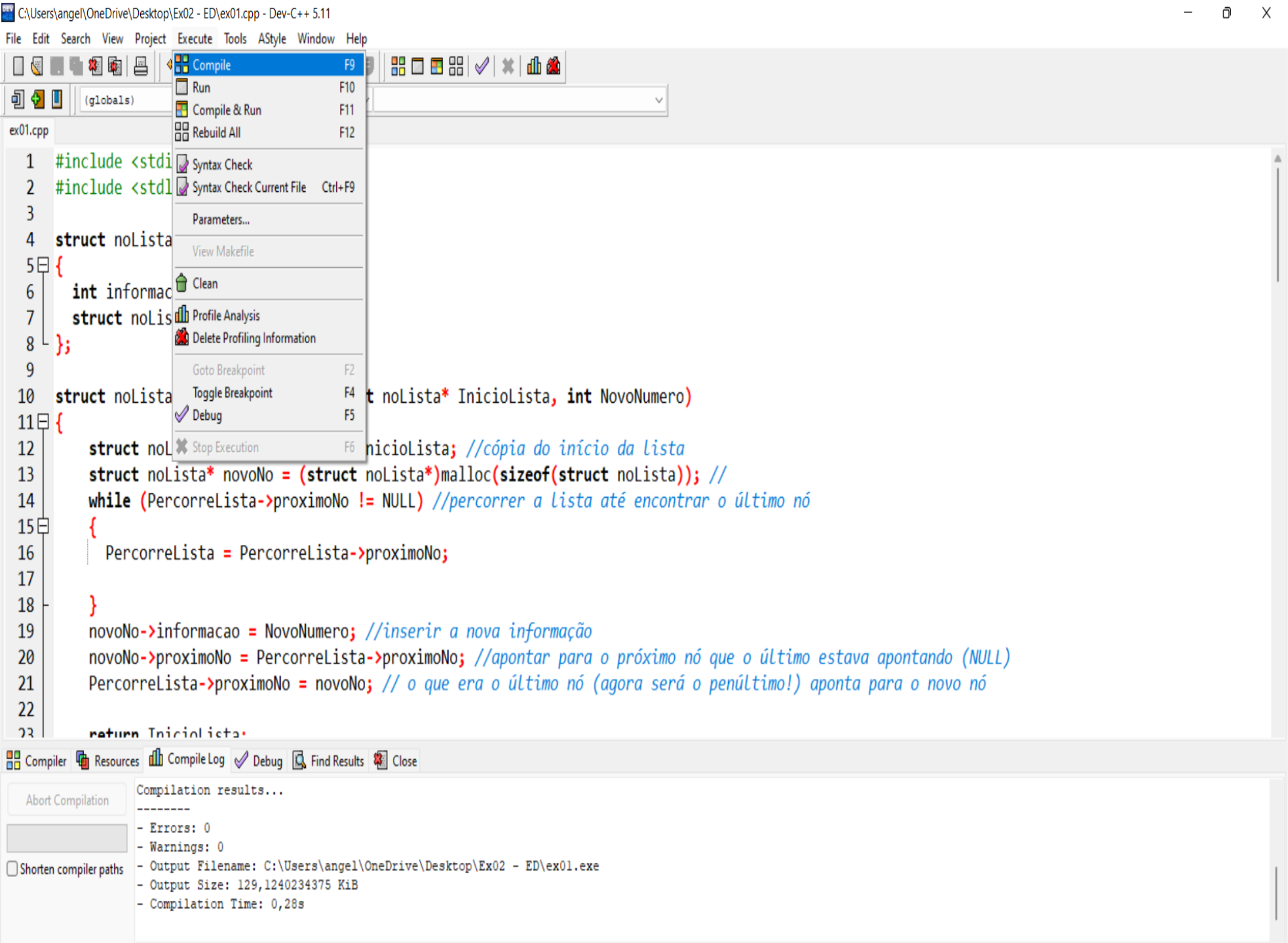
C:\Users\angel\OneDrive\Desktop\Ex02 - ED\ex01.cpp - Dev-C++ 5.11

Edit Search View Project Execute Tools AStyle Window Help

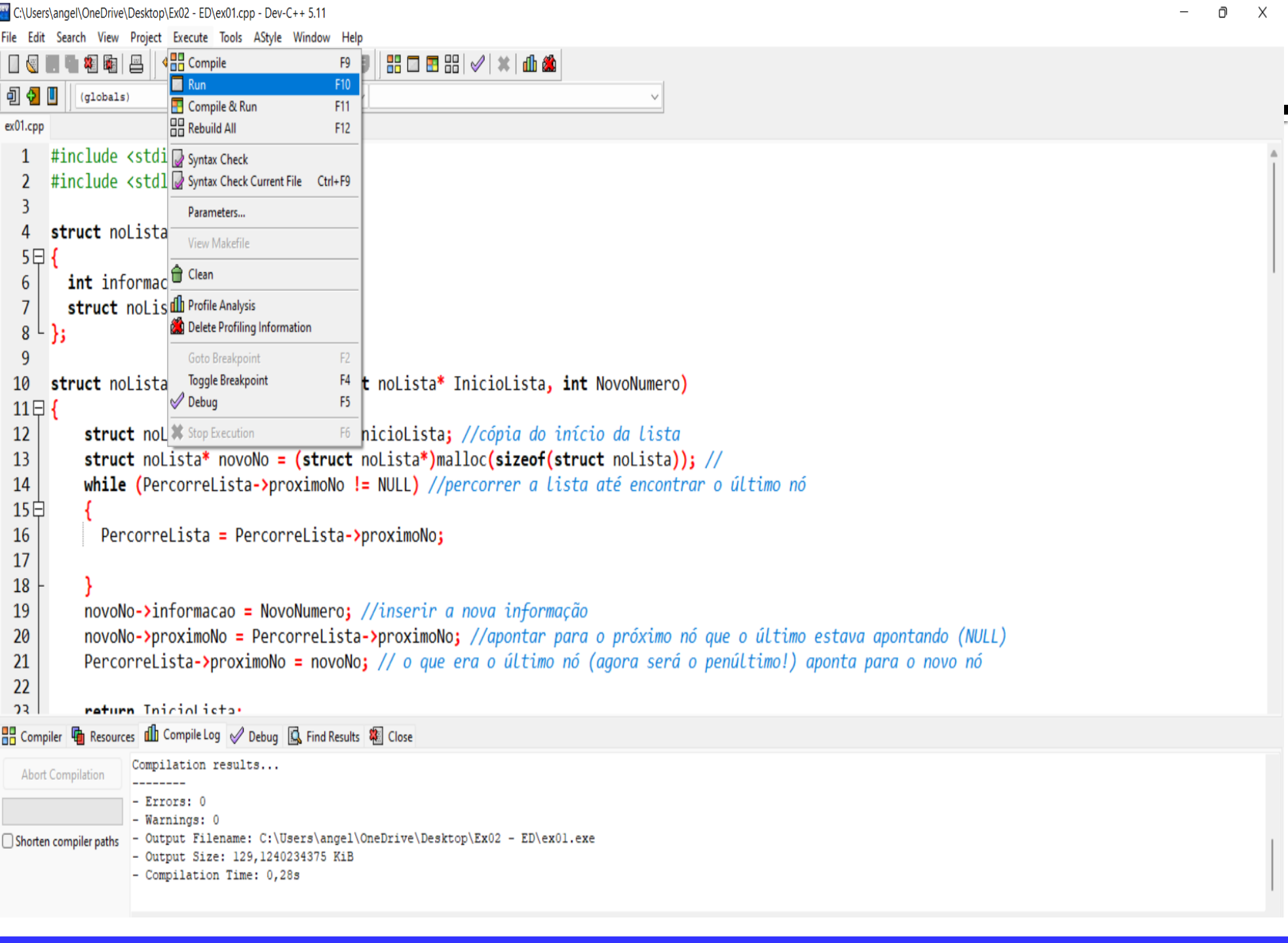
(globals)

x01.cpp

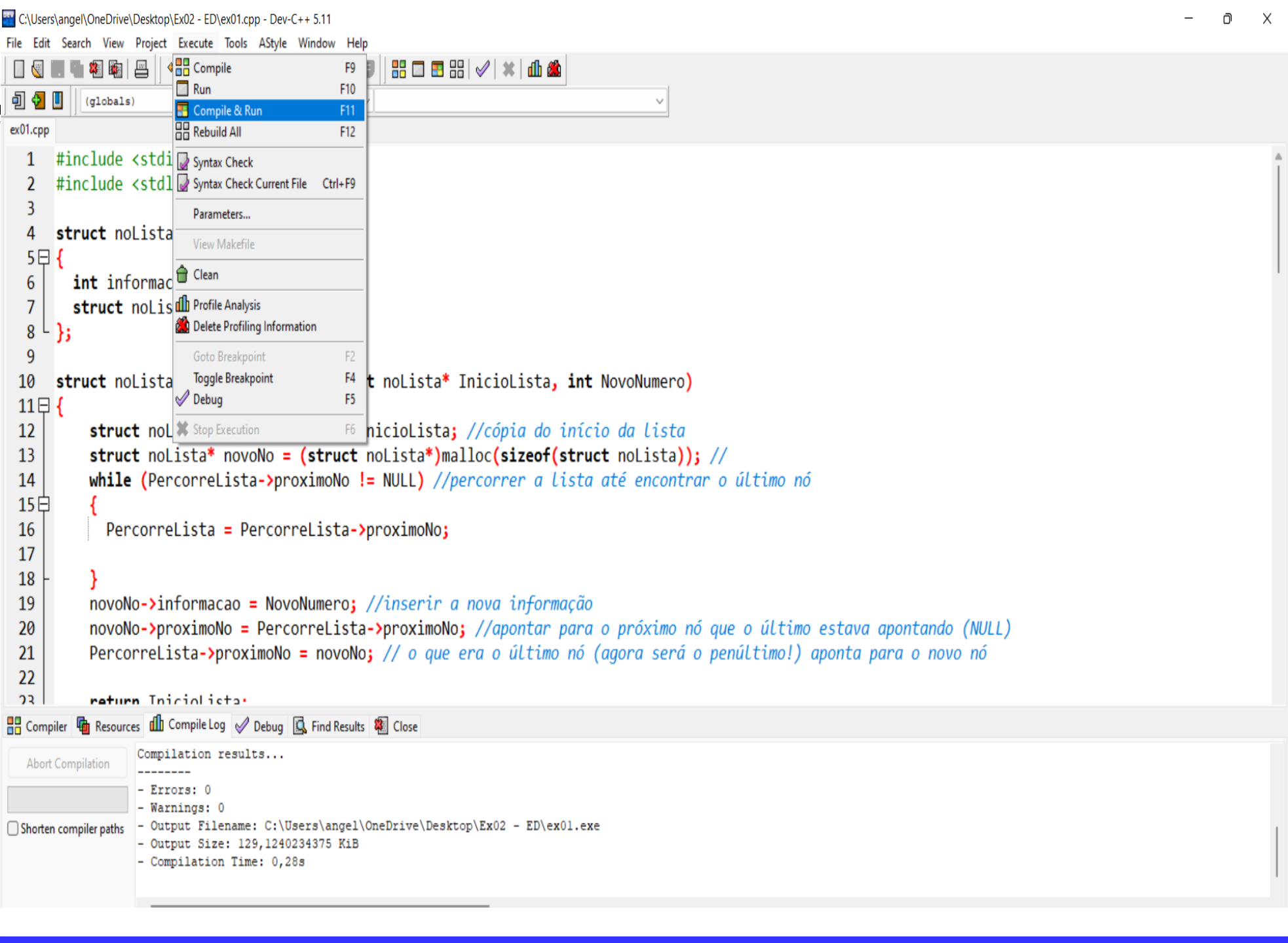
```c
#include <stdio.h>
#include <stdlib.h>

struct noLista
{
  int informacao;
  struct noLista *proximoNo;
};

struct noLista* inserirFimLista(struct noLista* InicioLista, int NovoNumero)
{
    struct noLista *PercorreLista = InicioLista; //cópia do início da lista
    struct noLista* novoNo = (struct noLista*)malloc(sizeof(struct noLista)); //
    while (PercorreLista->proximoNo != NULL) //percorrer a lista até encontrar o último nó
    {
      PercorreLista = PercorreLista->proximoNo;

    }
    novoNo->informacao = NovoNumero; //inserir a nova informação
    novoNo->proximoNo = PercorreLista->proximoNo; //apontar para o próximo nó que o último estava apontando (NULL)
    PercorreLista->proximoNo = novoNo; // o que era o último nó (agora será o penúltimo!) aponta para o novo nó

    return InicioLista;
}


void ImprimirLista(struct noLista  *InicioDaLista)
{
    noLista *NoAtual = InicioDaLista; //copiar o endereço do primeiro nó da lista
```


C:\Users\angel\OneDrive\Desktop\Ex02 - ED\ex01.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
Compile F9
Run F10
Compile & Run F11
Rebuild All F12
Syntax Check
Syntax Check Current File Ctrl+F9
Parameters...
View Makefile
Clean
Profile Analysis
Delete Profiling Information
Goto Breakpoint F2
Toggle Breakpoint F4
Debug F5
Stop Execution F6
(globals)
ex01.cpp
#include <stdi
#include <stdl
struct noLista
{
int informac
struct noLis
};
struct noLista t noLista* InicioLista, int NovoNumero)
{
struct noL nicioLista; //cópia do início da lista
struct noLista* novoNo = (struct noLista*)malloc(sizeof(struct noLista)); //
while (PercorreLista->proximoNo != NULL) //percorrer a lista até encontrar o último nó
{
PercorreLista = PercorreLista->proximoNo;
}
novoNo->informacao = NovoNumero; //inserir a nova informação
novoNo->proximoNo = PercorreLista->proximoNo; //apontar para o próximo nó que o último estava apontando (NULL)
PercorreLista->proximoNo = novoNo; // o que era o último nó (agora será o penúltimo!) aponta para o novo nó
return InicioLista;
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Shorten compiler paths
Compilation results...

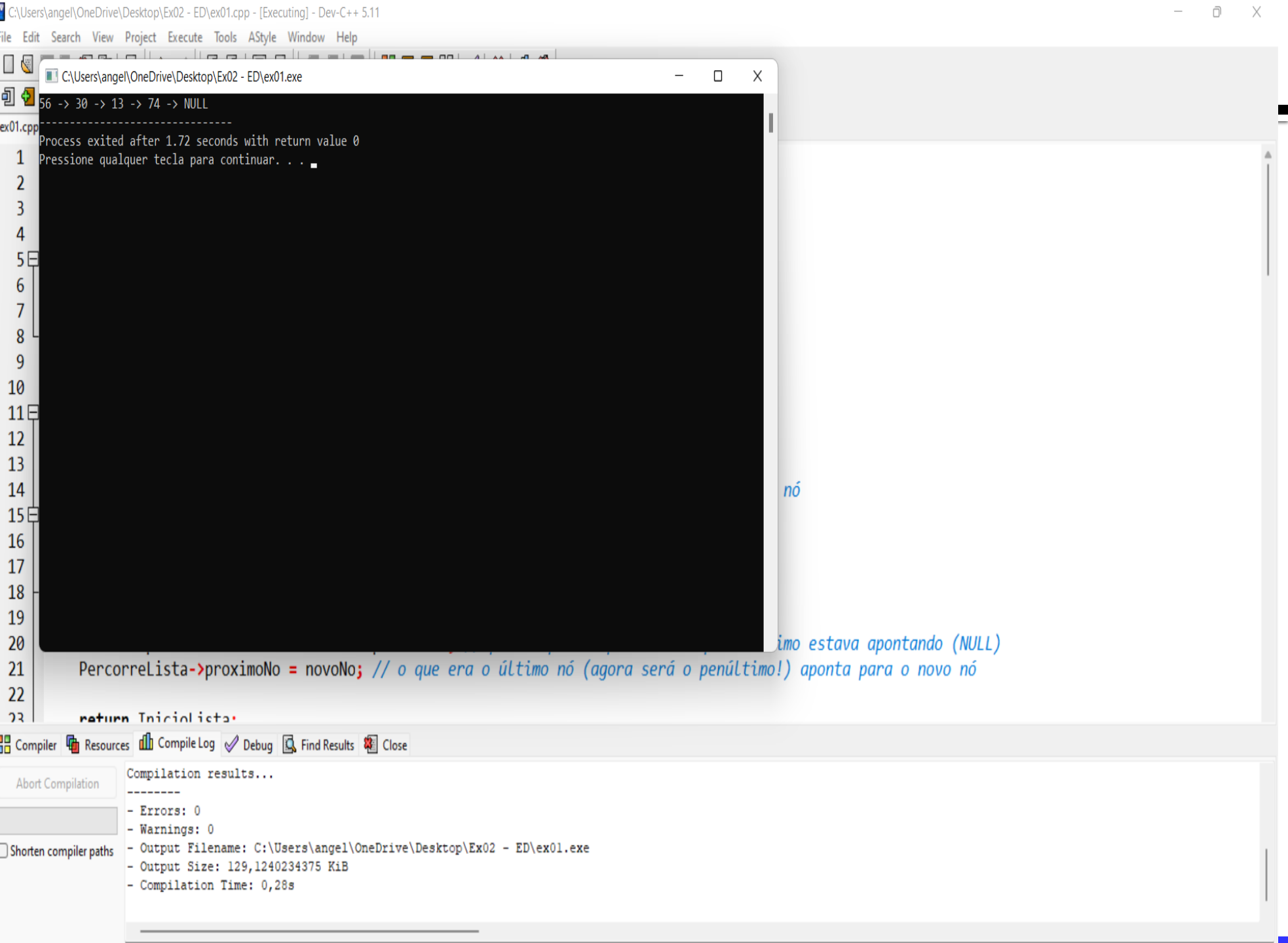
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\angel\OneDrive\Desktop\Ex02 - ED\ex01.exe
- Output Size: 129,1240234375 KiB
- Compilation Time: 0,28s


C:\Users\angel\OneDrive\Desktop\Ex02 - ED\ex01.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
Compile F9
Run F10
Compile & Run F11
Rebuild All F12
Syntax Check
Syntax Check Current File Ctrl+F9
Parameters...
View Makefile
Clean
Profile Analysis
Delete Profiling Information
Goto Breakpoint F2
Toggle Breakpoint F4
Debug F5
Stop Execution F6
(globals)
ex01.cpp
#include <stdi
#include <stdl
struct noLista
{
int informac
struct noLis
};
struct noLista t noLista* InicioLista, int NovoNumero)
{
struct noL nicioLista; //cópia do início da lista
struct noLista* novoNo = (struct noLista*)malloc(sizeof(struct noLista)); //
while (PercorreLista->proximoNo != NULL) //percorrer a lista até encontrar o último nó
{
PercorreLista = PercorreLista->proximoNo;
}
novoNo->informacao = NovoNumero; //inserir a nova informação
novoNo->proximoNo = PercorreLista->proximoNo; //apontar para o próximo nó que o último estava apontando (NULL)
PercorreLista->proximoNo = novoNo; // o que era o último nó (agora será o penúltimo!) aponta para o novo nó
return InicioLista;
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Shorten compiler paths
Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\angel\OneDrive\Desktop\Ex02 - ED\ex01.exe
- Output Size: 129,1240234375 KiB
- Compilation Time: 0,28s


C:\Users\angel\OneDrive\Desktop\Ex02 - ED\ex01.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
Compile F9
Run F10
Compile & Run F11
Rebuild All F12
Syntax Check
Syntax Check Current File Ctrl+F9
Parameters...
View Makefile
Clean
Profile Analysis
Delete Profiling Information
Goto Breakpoint F2
Toggle Breakpoint F4
Debug F5
Stop Execution F6
(globals)
ex01.cpp
#include <stdi
#include <stdl
struct noLista
{
int informac
struct noLis
};
struct noLista t noLista* InicioLista, int NovoNumero)
{
struct noL nicioLista; //cópia do início da lista
struct noLista* novoNo = (struct noLista*)malloc(sizeof(struct noLista)); //
while (PercorreLista->proximoNo != NULL) //percorrer a lista até encontrar o último nó
{
PercorreLista = PercorreLista->proximoNo;
}
novoNo->informacao = NovoNumero; //inserir a nova informação
novoNo->proximoNo = PercorreLista->proximoNo; //apontar para o próximo nó que o último estava apontando (NULL)
PercorreLista->proximoNo = novoNo; // o que era o último nó (agora será o penúltimo!) aponta para o novo nó
return InicioLista;
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Shorten compiler paths
Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\angel\OneDrive\Desktop\Ex02 - ED\ex01.exe
- Output Size: 129,1240234375 KiB
- Compilation Time: 0,28s


C:\Users\angel\OneDrive\Desktop\Ex02 - ED\ex01.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
C:\Users\angel\OneDrive\Desktop\Ex02 - ED\ex01.exe
56 -> 30 -> 13 -> 74 -> NULL

Process exited after 1.72 seconds with return value 0
Pressione qualquer tecla para continuar. . .
nó
imo estava apontando (NULL)
PercorreLista->proximoNo = novoNo; // o que era o último nó (agora será o penúltimo!) aponta para o novo nó
return InicioLista;
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Shorten compiler paths
Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\angel\OneDrive\Desktop\Ex02 - ED\ex01.exe
- Output Size: 129,1240234375 KiB
- Compilation Time: 0,28s

Exercícios 😀

1- Implemente uma estrutura de um nó de uma lista encadeada para armazenar a idade de uma pessoa.

2- Implemente uma função para inserir um nó no início da lista encadeada.

3- Implemente uma função para imprimir todos os elementos de uma lista encadeada.

4- No programa principal, inicie o Primeiro nó da lista com NULL e insira 5 idades no início da lista. Obs.: solicite para o usuário digitar uma idade por vez!

5- Imprima a lista encadeada com as 5 idades que foram digitadas.

6- Implemente uma função para inserir um nó no final da lista encadeada.

7- No programa principal, insira 2 idades no final da lista. Obs.: solicite para o usuário digitar uma idade por vez!

7- No programa principal, insira 2 idades no final da lista. Obs.: solicite para o usuário digitar uma idade por vez!

8- Imprima a lista encadeada contendo as 7 idades digitadas.

9- Implemente uma função para remover uma determinada idade da lista encadeada.

10- No programa principal, remova uma determinada idade da lista encadeada. Obs.: solicite para o usuário digitar uma idade!

11- Imprima a lista encadeada atualizada, após a remoção da idade.

12- Implemente uma função para procurar uma determinada idade em uma lista encadeada. A função deve retornar o nó contendo a idade encontrada ou NULL caso a idade procurada não esteja na lista.

13- No programa principal, solicite para o usuário digitar uma idade para ser procurada na lista. Exiba uma mensagem dizendo se o número está ou não na lista. Em seguida, imprima todo o conteúdo da lista.

14 – Implemente uma função para inserir uma idade em uma posição específica de uma lista encadeada.

15 - No programa principal, solicite que o usuário digite a idade a posição em que ela deve ser inserida na lista encadeada. Imprima o resultado da lista.

16- Implemente uma função para procurar uma determinada idade em uma lista encadeada e retornar a POSIÇÃO em que esta idade está na lista. Caso a idade não esteja na lista, a função deve retornar -1.

17- Implemente uma função que insira idades em uma lista encadeada de forma ORDENADA CRESCENTE.

18 - No programa principal, insira as idades digitadas pelo usuário utilizando a função de inserção ORDENADA CRESCENTE. Imprima o resultado da lista.

19 – Faça uma função que percorra a lista encadeada e imprima apenas os números pares.

20 Implemente uma função que insira idades em uma lista encadeada de forma ORDENADA DESCRESCENTE.

Muito Obrigada!

Profª. Angela Abreu Rosa de Sá, Drª.

Contato: angelaabreu@gmail.com