

INSTITUTO MAUÁ DE TECNOLOGIA



13

Pilhas e Bibliotecas

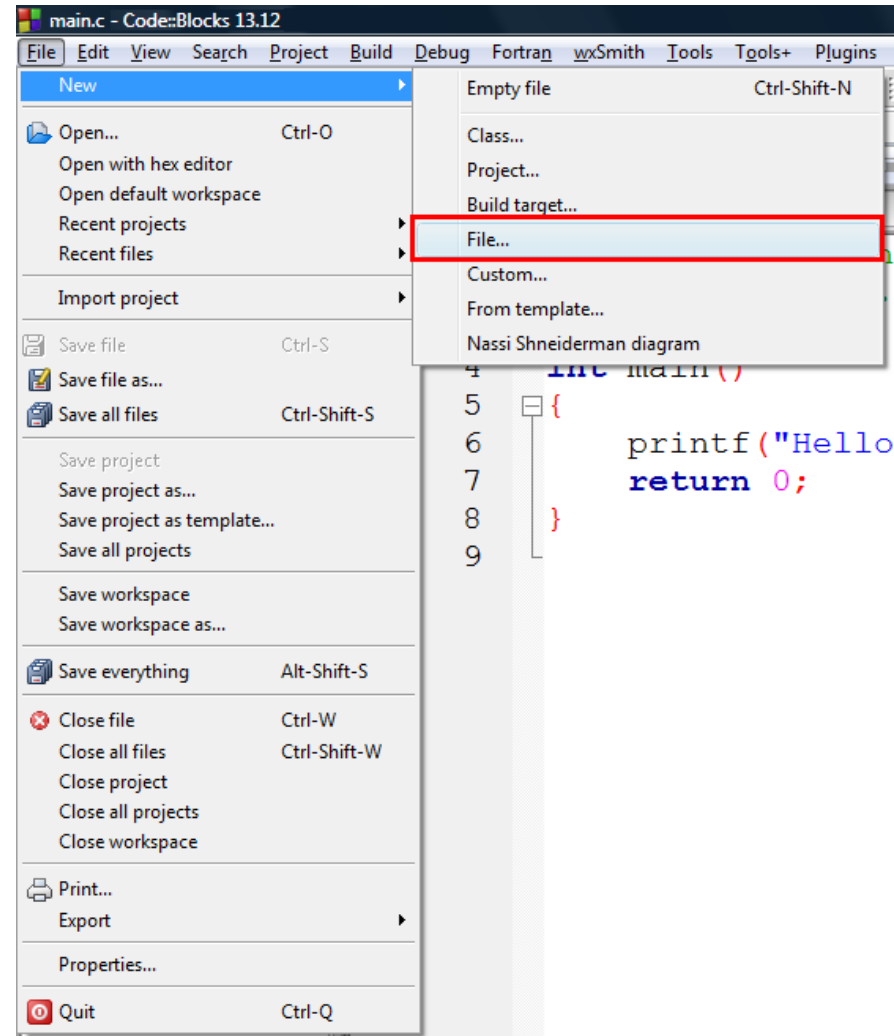


Bibliotecas

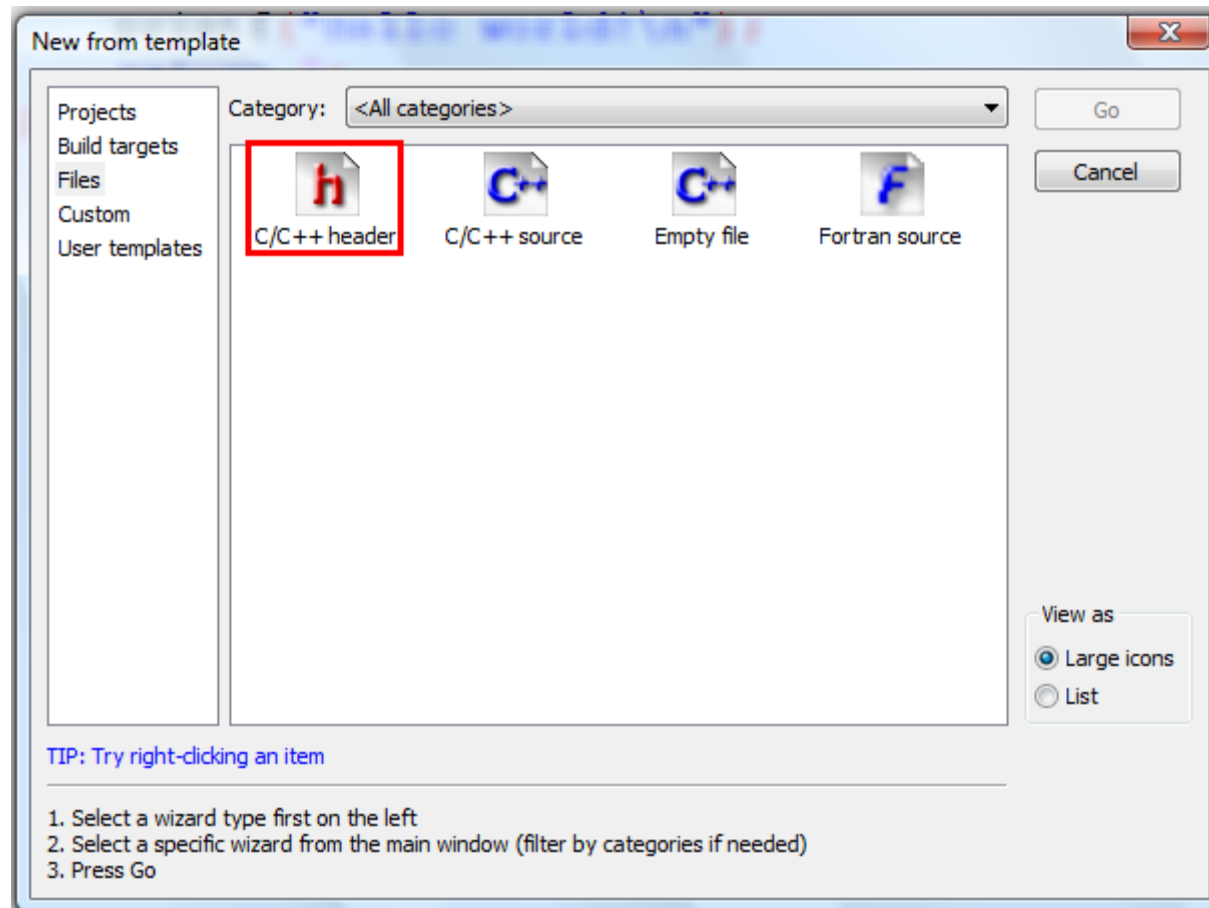
- São arquivos que contém estruturas de dados e sub-rotinas destinadas à resolução de um determinado problema;
- Muito utilizadas em Tipos Abstratos de Dados;
- São criados dois arquivos:
 - **Arquivo de código (.c):**
 - código-fonte das sub-rotinas;
 - incluir o arquivo de cabeçalho utilizando:
#include “cabeçalho.h”
 - **Arquivo de cabeçalho (.h):**
 - estruturas de dados;
 - cabeçalhos das sub-rotinas que serão disponibilizadas.

Bibliotecas

Após criar um projeto novo, criar o arquivo.h

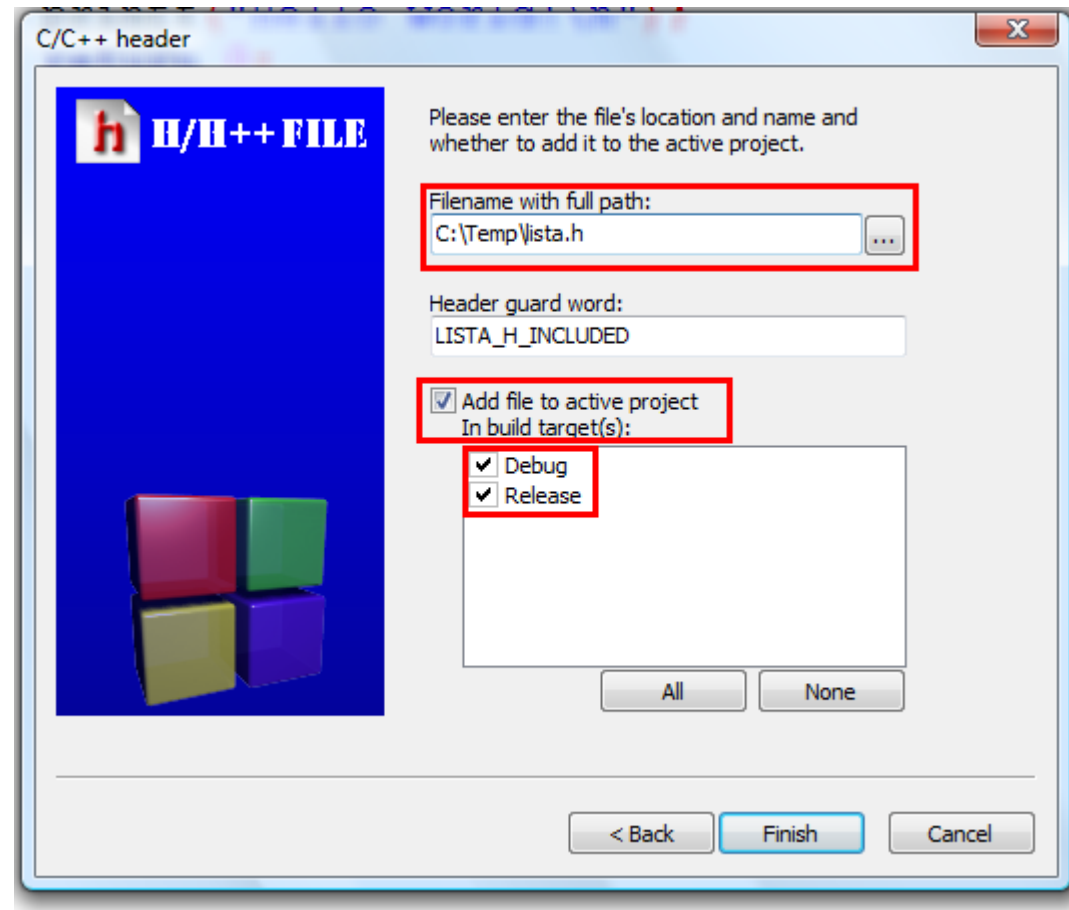


Bibliotecas



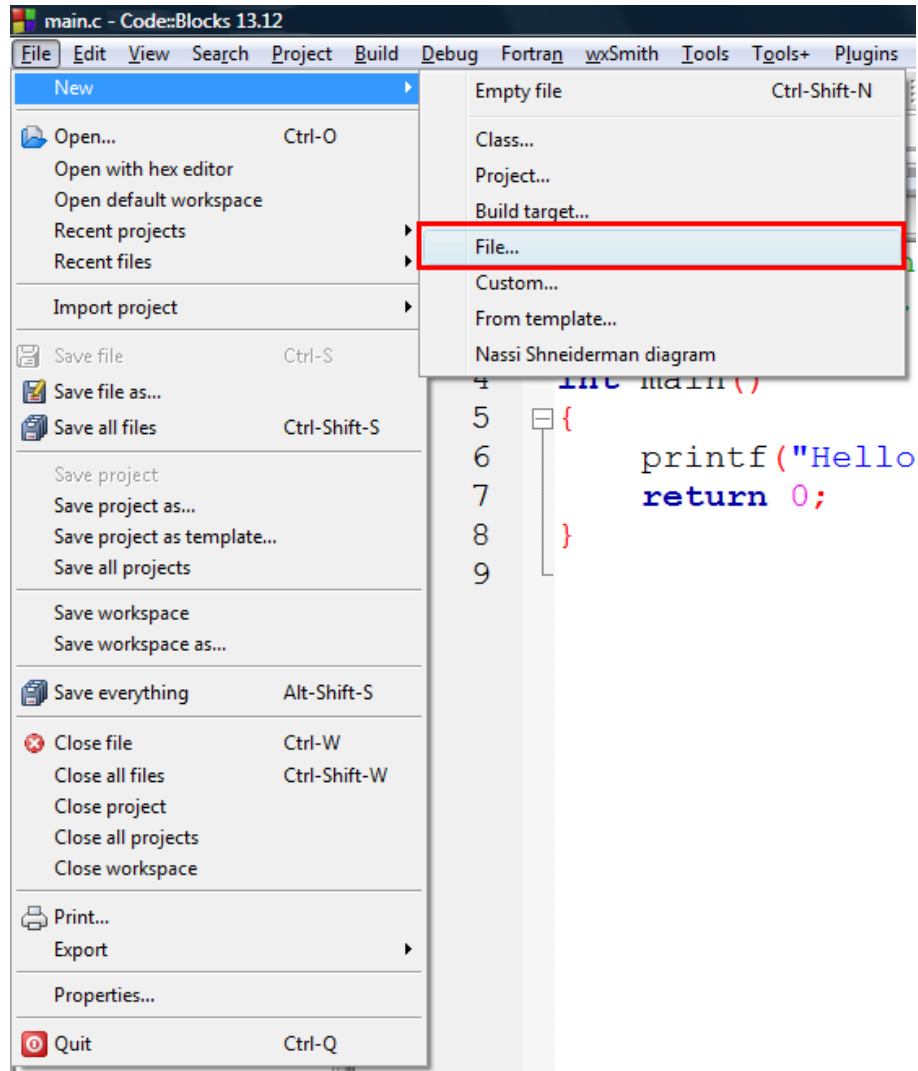
Bibliotecas

Salve o arquivo lista.h na mesma pasta onde o projeto foi salvo!

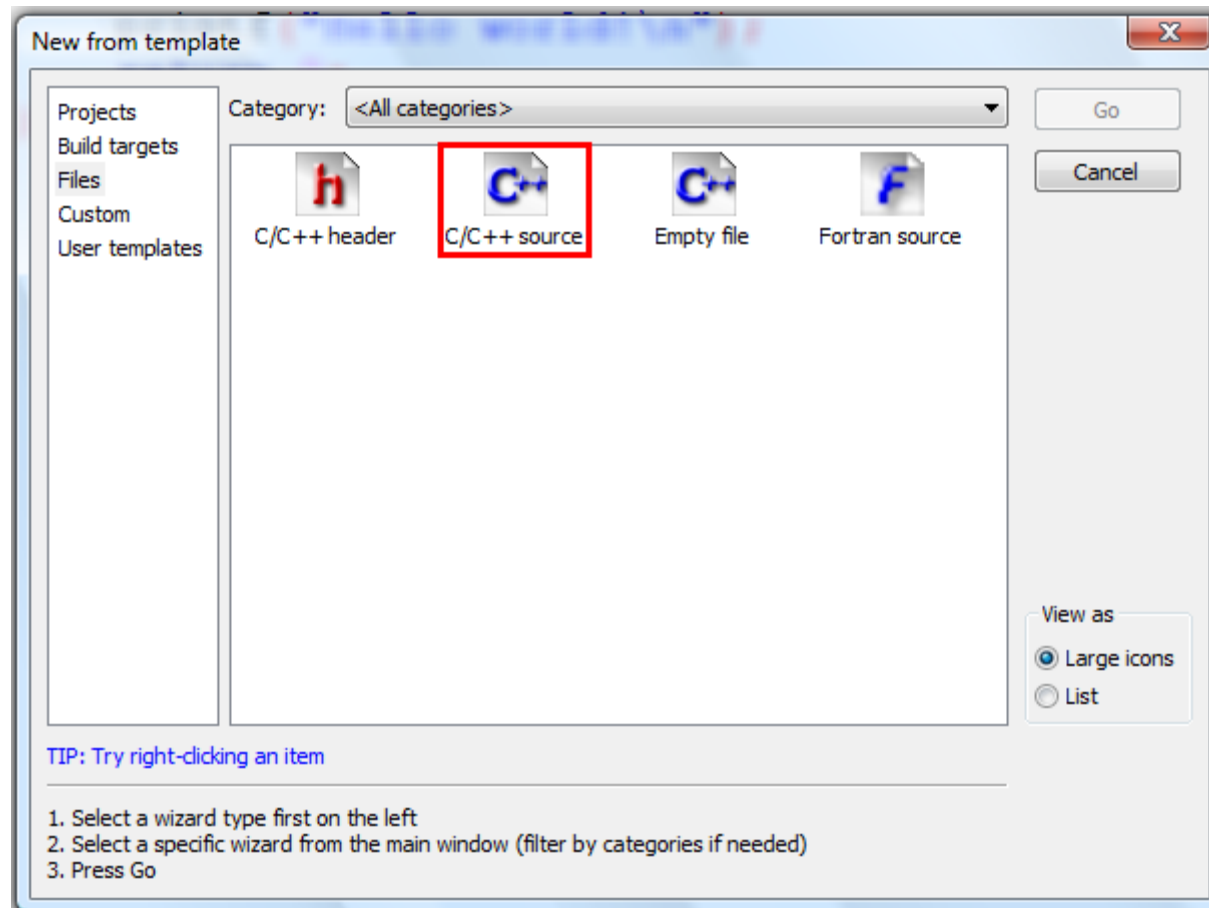


Bibliotecas

Agora vamos criar o arquivo.c



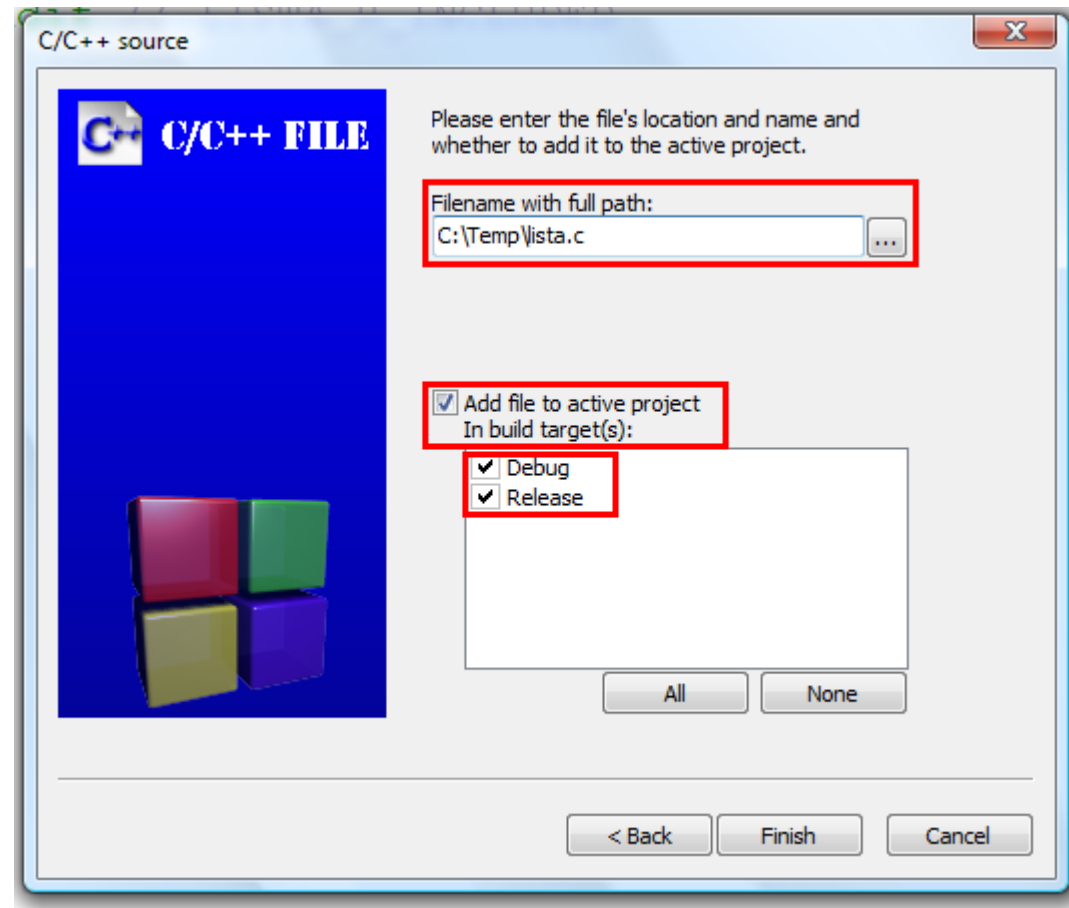
Bibliotecas



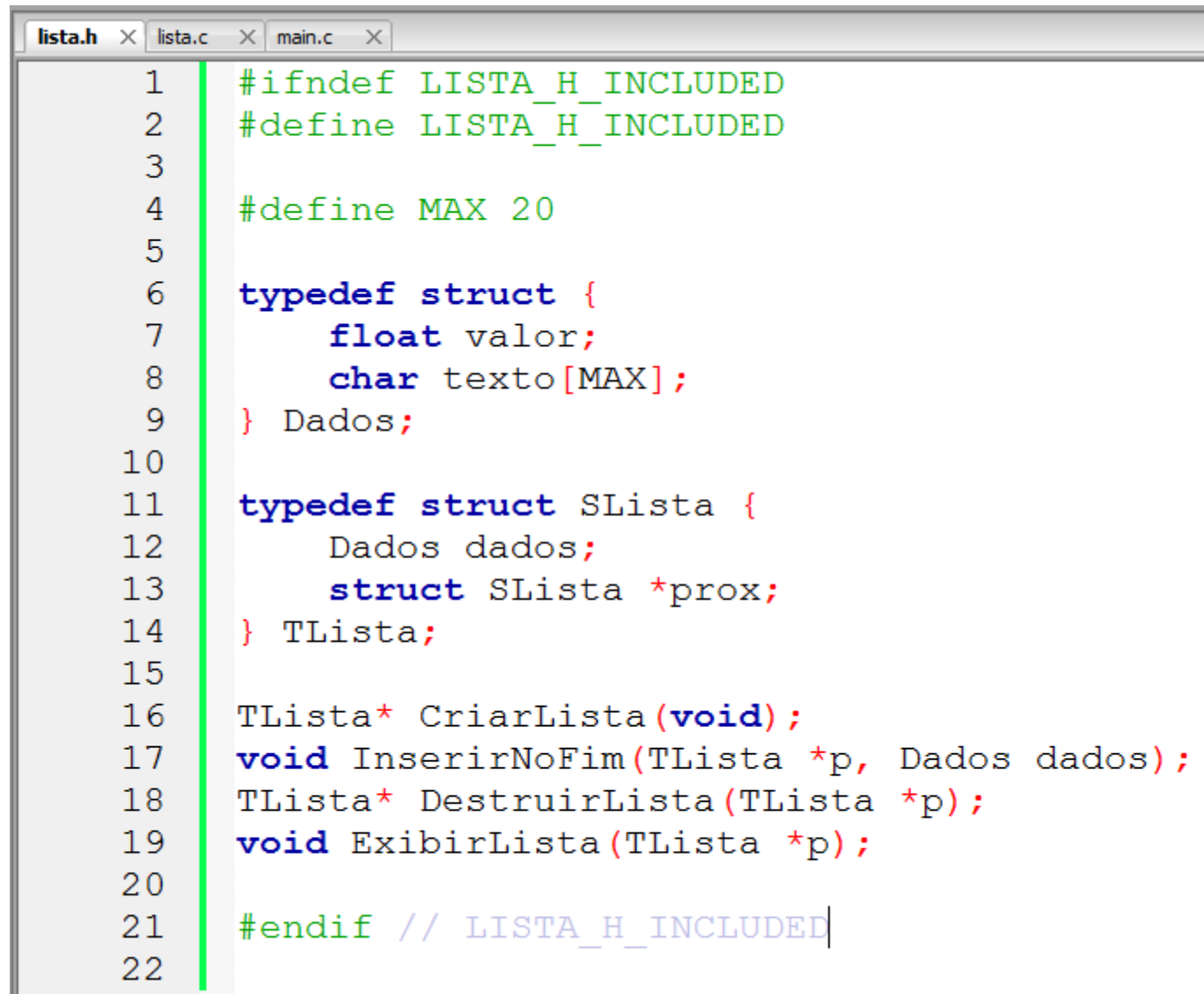
Bibliotecas

Salve o arquivo lista.c na mesma pasta onde o projeto e o arquivo lista.h foram salvos!

Normalmente os arquivos .h e .c tem o mesmo nome.



Bibliotecas



```
1  #ifndef LISTA_H_INCLUDED
2  #define LISTA_H_INCLUDED
3
4  #define MAX 20
5
6  typedef struct {
7      float valor;
8      char texto[MAX];
9  } Dados;
10
11 typedef struct SLista {
12     Dados dados;
13     struct SLista *prox;
14 } TLista;
15
16 TLista* CriarLista(void);
17 void InserirNoFim(TLista *p, Dados dados);
18 TLista* DestruirLista(TLista *p);
19 void ExibirLista(TLista *p);
20
21 #endif // LISTA_H_INCLUDED
22
```

Bibliotecas

Complete o código com as quatro rotinas apresentadas.

```
lista.h x lista.c x main.c x
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include "lista.h"
4
5  TLista* CriarLista(void)
6  {
7      TLista *p;    /* Ponteiro para a lista */
8
9      p = (TLista*) malloc(sizeof(TLista));
10     if (p == NULL) {
11         printf("Não pode criar a lista");
12         exit(EXIT_FAILURE);
13     } else {
14         p->prox = NULL;    /* Atribui ponteiro nulo */
15         return p;    /* Retorna endereço da lista */
16     }
17 }
18
19 void InserirNoFim(TLista *p, Dados dados) {
20     TLista* novo;
21     novo = (TLista*) malloc(sizeof(TLista));
22
23     if (novo == NULL) {
24         printf("Nao foi possivel alocar memoria!");
25         exit(EXIT_FAILURE);
26     } else {
27         novo->dados = dados;
```

Bibliotecas

```
lista.h x lista.c x main.c x
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include "lista.h"
4
5  void LerString(char s[MAX]) {
6      int t;
7      char c;
8      /* limpa o buffer */
9      while((c = getchar()) != '\n' && c != EOF);
10     fgets(s, MAX, stdin);
11     t = strlen(s);
12     if(s[t-1] == '\n')
13         s[t-1] = '\0';
14 }
15
16 int main() {
17     char op;
18     Dados dados;
19     TLista *lista;
20     lista = CriarLista();
21     do {
22         printf("valor: ");
23         scanf("%f", &dados.valor);
24         printf("texto: ");
25         LerString(dados.texto);
26         InserirNoFim(lista, dados);
27         printf("Mais itens (S/N)? ");
```

Pilhas

Pilha (*stack*)

Empilha elementos que deverão ser processados na ordem “primeiro que entra é o último que sai” (exemplo: pilha de pratos);

Em computação, uma pilha serve para:

- armazenar o estado do programa em chamadas recursivas (lembra do QuickSort?);
- representar processadores reais ou virtuais baseados em pilha (a calculadora HP é baseada em pilha);
- auxílio na avaliação de expressões algébricas;
- aplicações específicas em alguns algoritmos (busca em profundidade, etc.).

Pilha (*stack*)

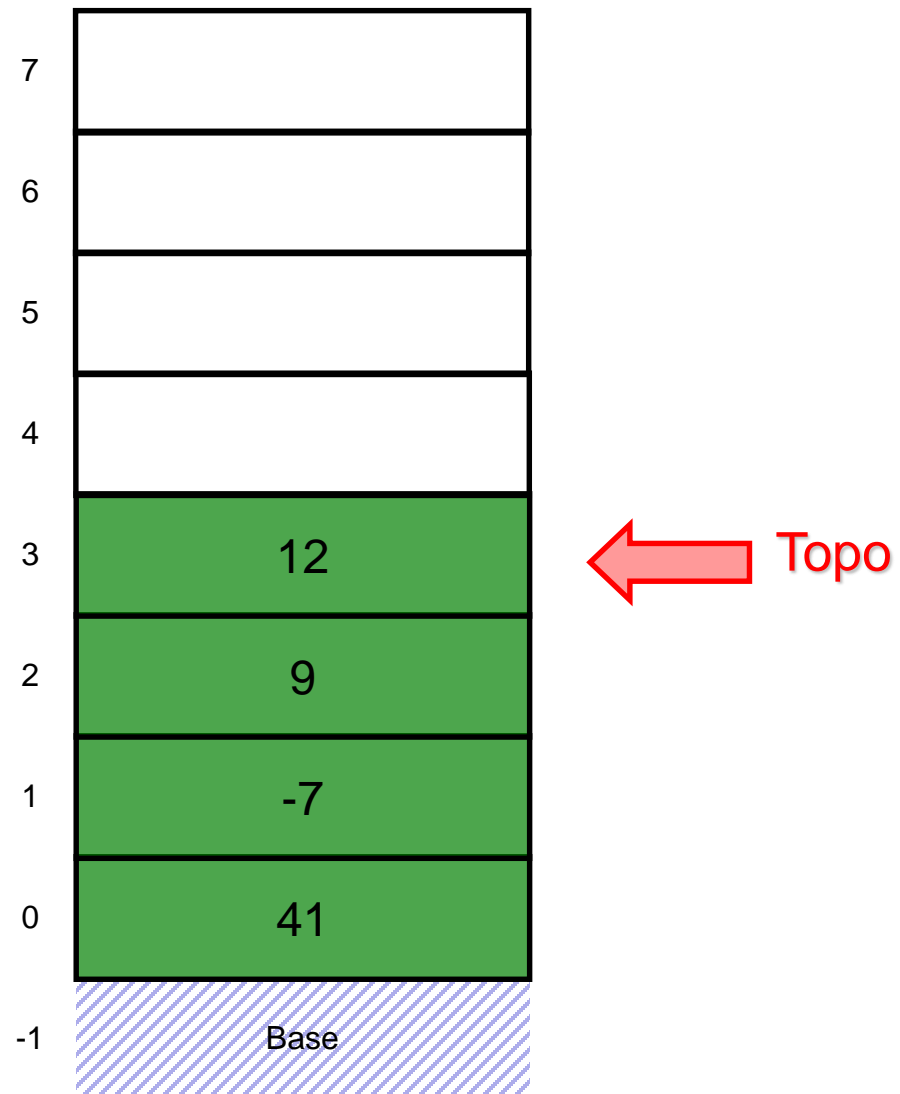
LIFO: Last In First Out;

a operação de inserção **coloca um elemento no topo** da pilha enquanto que a operação de retirada **retira o elemento que estiver no topo**;

Estudaremos uma pilha implementada com um vetor:

- itens são inseridos na posição “topo”;
- itens são retirados na posição “topo”;
- a pilha está vazia quando o topo é -1 (ou a base);
- a pilha está cheia quando o topo é o tamanho total da pilha.

Pilha (*stack*)



pilha.h

```
#ifndef PILHA_H_INCLUDED
#define PILHA_H_INCLUDED

#define MAX 20
#define MAX_PILHA 10

typedef struct {
    float valor;
    char texto[MAX];
} Dados;

typedef struct {
    Dados elemento[MAX_PILHA];
    int topo;
} TPilha;
```

pilha.h

```
//cria uma pilha nova

void CriarPilha(TPilha *p);

//insere

int InserirNaPilha(TPilha *p, Dados dados);

//retira

int RetirarDaPilha(TPilha *p, Dados *dados);

//indica se a pilha está vazia

int PilhaVazia(TPilha p);

//indica se a pilha está cheia

int PilhaCheia(TPilha p);

//retorna o tamanho da pilha

int QuantidadeNaPilha(TPilha p);

#endif // PILHA_H_INCLUDED
```

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

void CriarPilha(TPilha *p) {
    /* define valores iniciais das propriedades da pilha */
    p->topo = -1;
}
```

```
int InserirNaPilha(TPilha *p, Dados dados) {  
    int pc = PilhaCheia(*p);  
    if (!pc) {  
        p->topo++;           /* incrementa posição do topo */  
        /* insere elemento no topo da pilha */  
        p->elemento[p->topo] = dados;  
    }  
    return !pc; /* retorna se conseguiu inserir o dado */  
}
```

pilha.c

```
int RetirarDaPilha(TPilha *p, Dados *dados) {  
    int pv;  
    pv = PilhaVazia(*p);  
    if (!pv) {  
        /* recupera informação do topo da pilha */  
        *dados = p->elemento[p->topo];  
        p->topo--;          /* decrementa posição do topo */  
    }  
    return !pv; /* retorna se conseguiu remover o dado */  
}
```

pilha.c

```
int PilhaVazia(TPilha p) {  
    return p.topo == -1;  
}  
  
int PilhaCheia(TPilha p) {  
    return p.topo == MAX_PILHA - 1;  
}  
  
int QuantidadeNaPilha(TPilha p) {  
    return p.topo + 1;  
}
```