MAUÁ ESCOLA DE ENGENHARIA

12

# Criação de Bibliotecas
# *Insertion Sort*

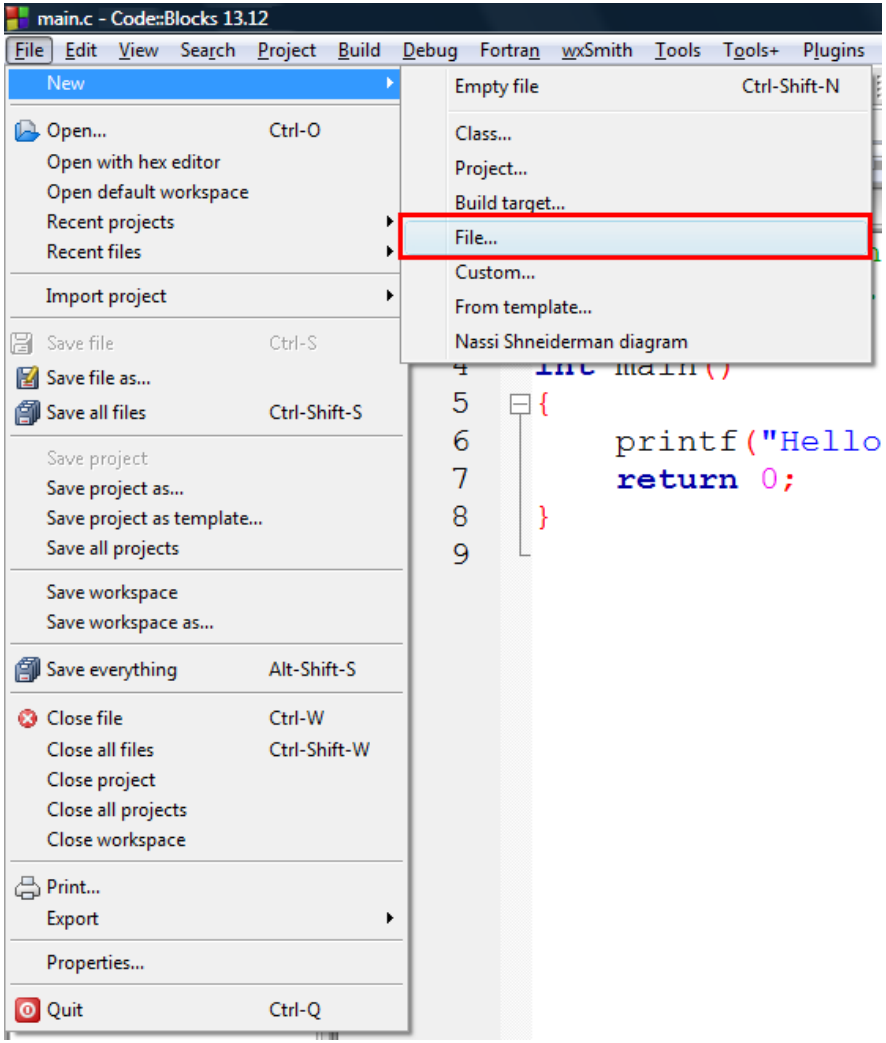
# Bibliotecas

- 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:

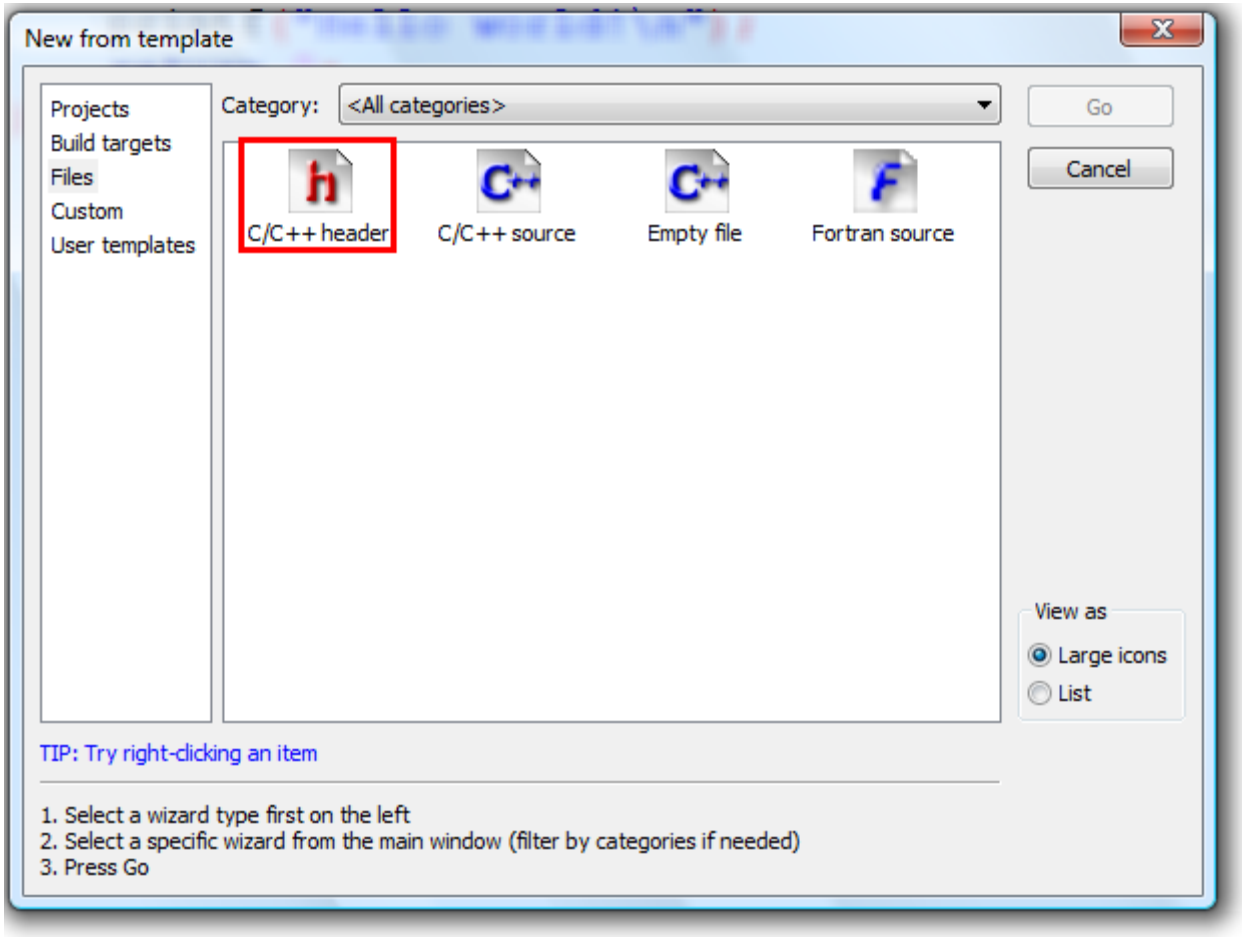
      **#include “cabeçalho.h”**
  - Arquivo de cabeçalho (.h):
    - estruturas de dados;
    - cabeçalhos das sub-rotinas que serão disponibilizadas.

# Bibliotecas

Após criar um projeto novo, criar o arquivo.h

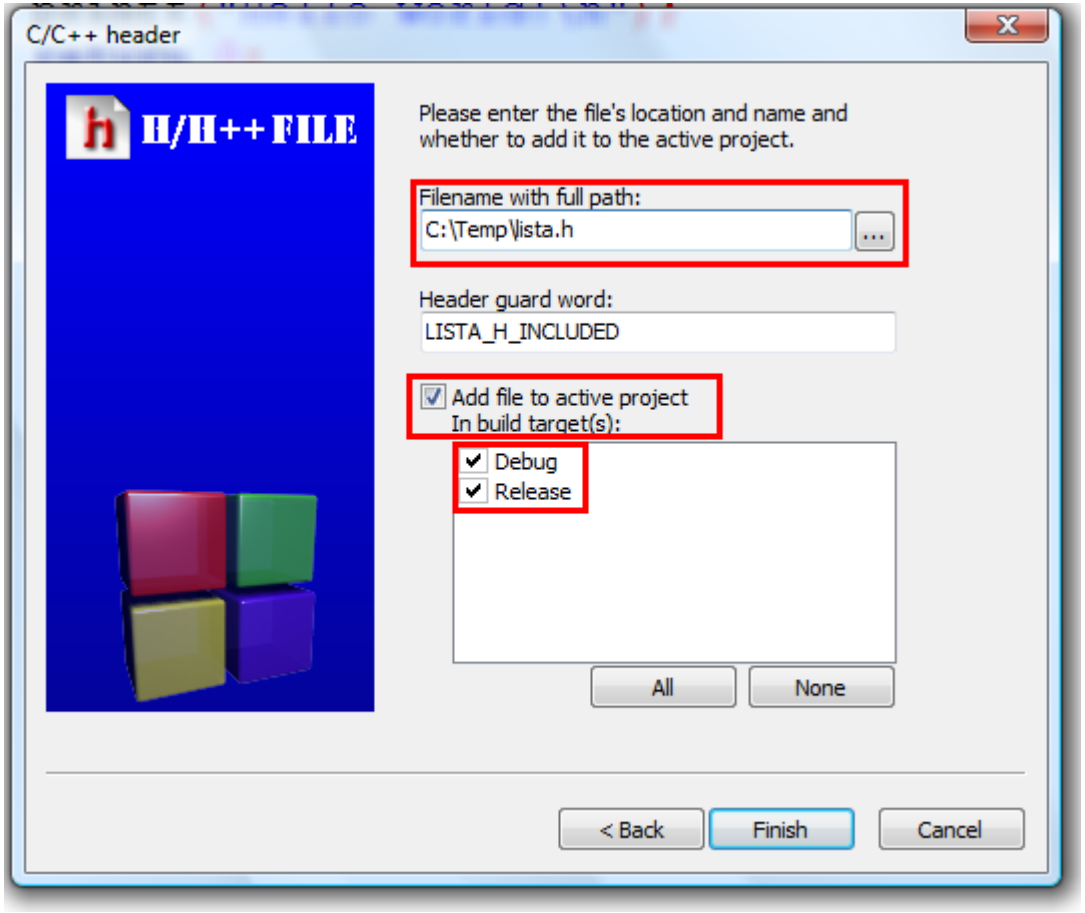


# Bibliotecas



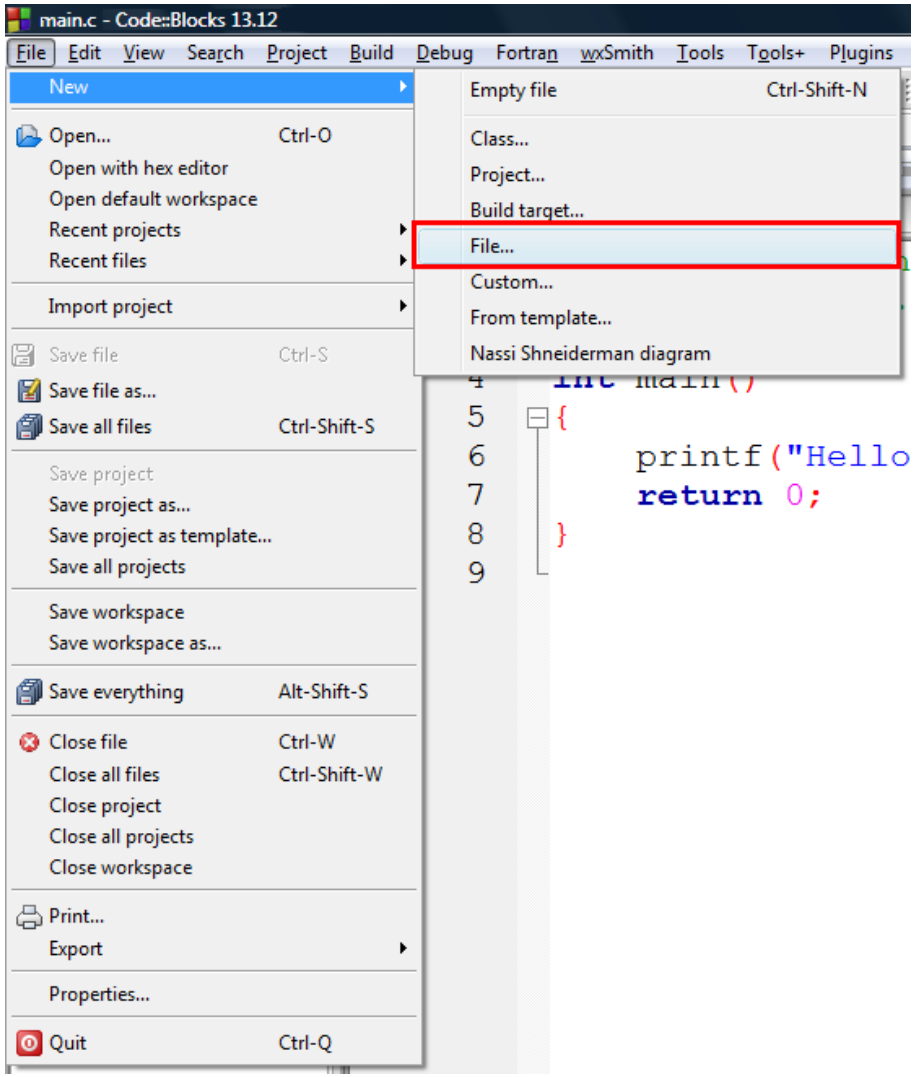
# Bibliotecas

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

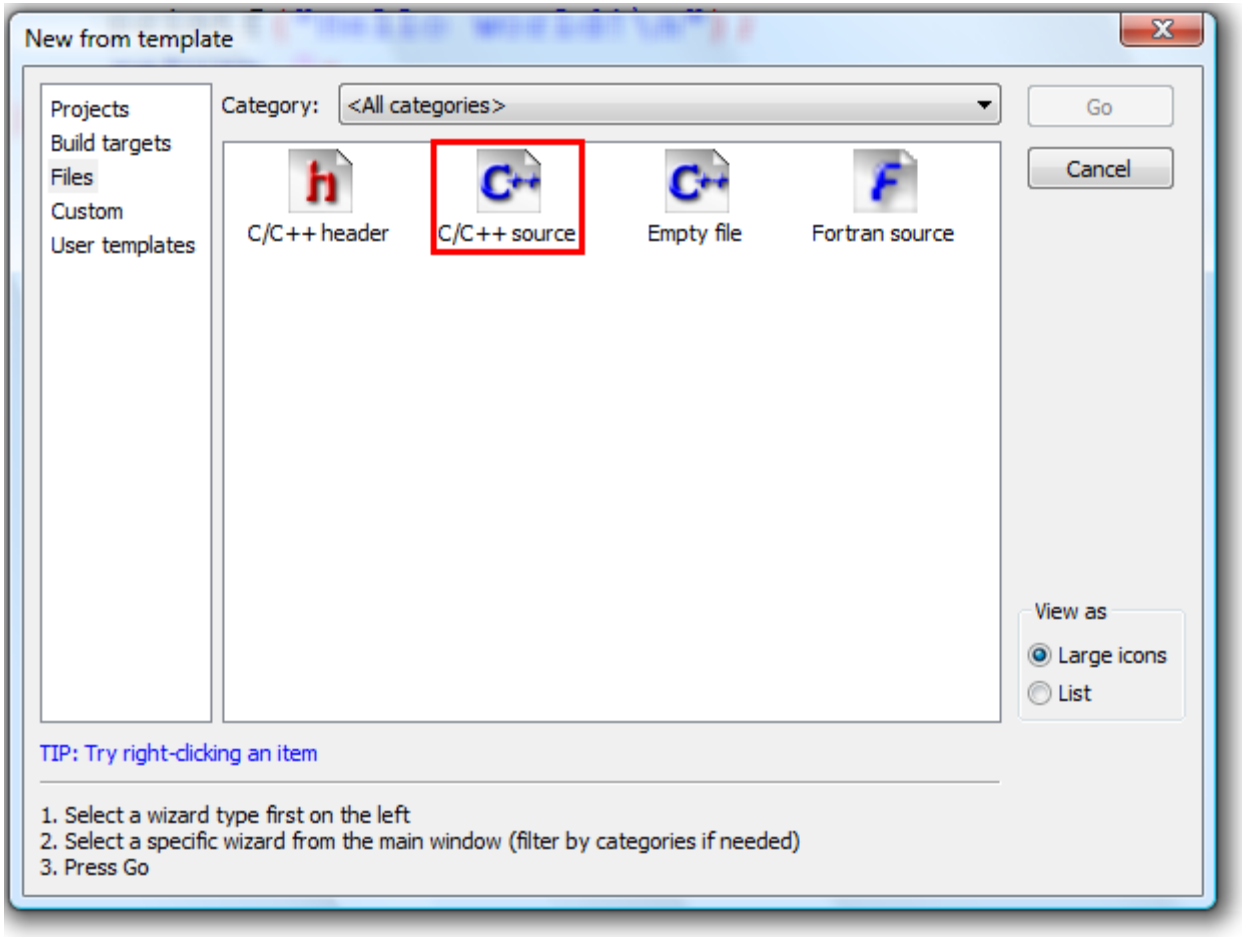


# Bibliotecas

Agora vamos criar o arquivo.c



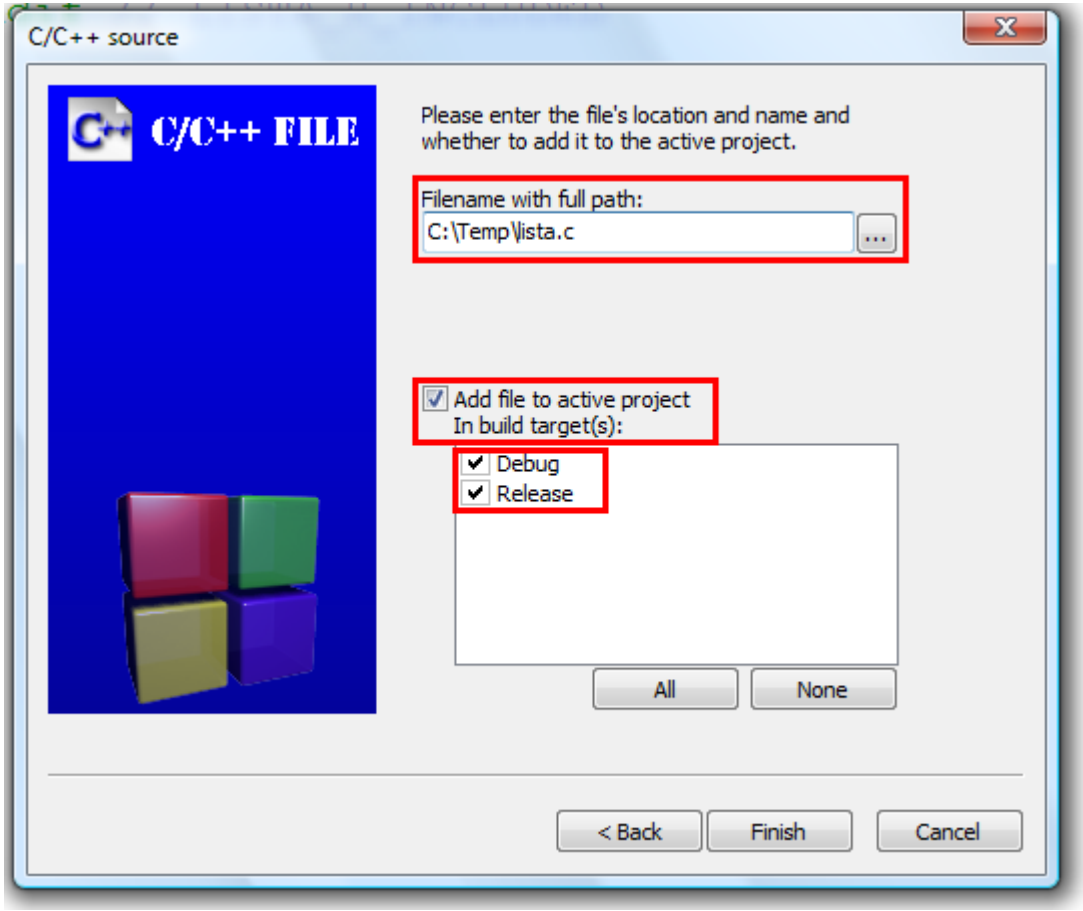
# Bibliotecas



# 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

lista.h | lista.c | main.c

```
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 | lista.c | main.c

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

TLista* CriarLista(void)
{
  TLista *p;    /* Ponteiro para a lista */

  p = (TLista*) malloc(sizeof(TLista));
  if (p == NULL){
    printf("Não pode criar a lista");
    exit(EXIT_FAILURE);
  } else {
    p->prox = NULL;    /* Atribui ponteiro nulo */
    return p;    /* Retorna endereço da lista */
  }
}

void InserirNoFim(TLista *p, Dados dados) {
    TLista* novo;
    novo = (TLista*) malloc(sizeof(TLista));

    if (novo == NULL) {
        printf("Nao foi possivel alocar memoria!");
        exit(EXIT_FAILURE);
    } else {
        novo->dados = dados;
```

# Bibliotecas

lista.h | lista.c | main.c

```
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)? ");
```

# *Insertion Sort*

- **Utilizado nas situações em que os dados chegam aos poucos;**
- **Considera-se que a lista de dados existentes já está ordenada;**
- **Os novos dados devem ser inseridos em sua correta posição, varrendo a lista desde o início até localizar onde o novo elemento será inserido.**

# Insertion Sort

- Fazer uma cópia da função InserirNoFim e renomeá-la para InserirEmOrdem no arquivo lista.c;
- Acrescentar o protótipo da nova função ao arquivo lista.h;
- No programa principal, substituir a função InserirNoFim pela InserirEmOrdem;
- Fazer as devidas alterações no código de InserirEmOrdem para que os novos dados (campos valor e texto) sejam inseridos na lista em ordem crescente do campo valor.