

CCP130

Desenvolvimento de

Algoritmos

Prof. Danilo H. Perico



Gerenciando grandes programas em C

Gerenciando grandes programas em C

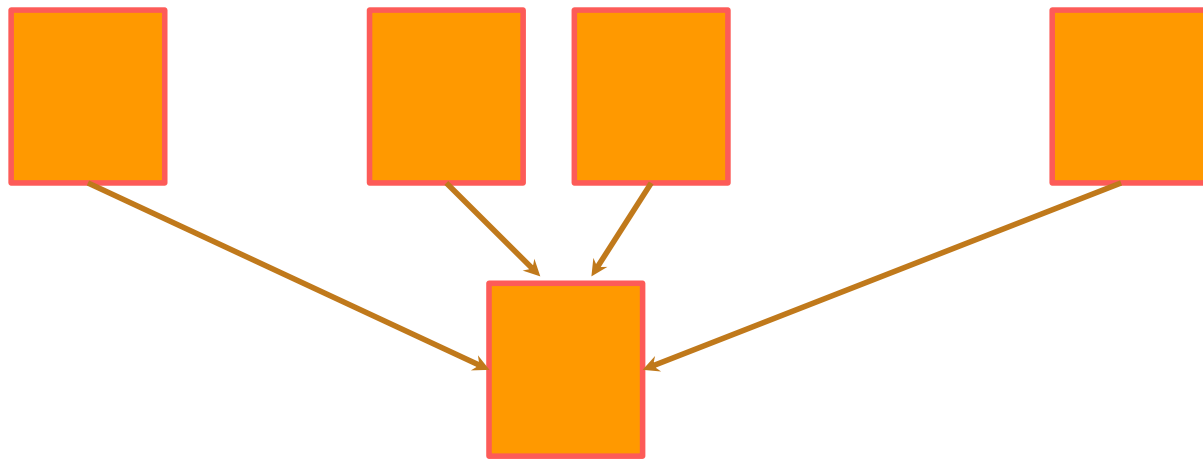
Todos os programas que escrevemos até agora estão em um único arquivo

Gerenciando grandes programas em C

- Mas e se eu tiver um **projeto enorme**, como o desenvolvimento de um jogo?
- Como ficaria este arquivo?
- E para dar manutenção nele?
- E para compilá-lo? Toda vez precisaria ser recompilado inteiramente?

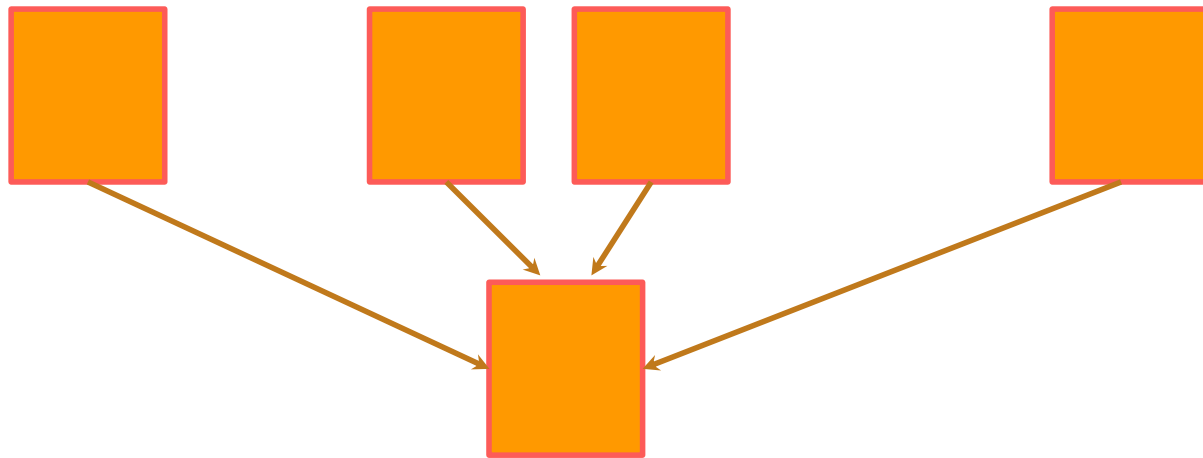
Gerenciando grandes programas em C

- Grandes programas, como jogos, ERPs (*Enterprise Resource Planning*), sistemas empresariais, etc. devem ser **divididos em arquivos menores** para manutenibilidade e compilação mais rápida.



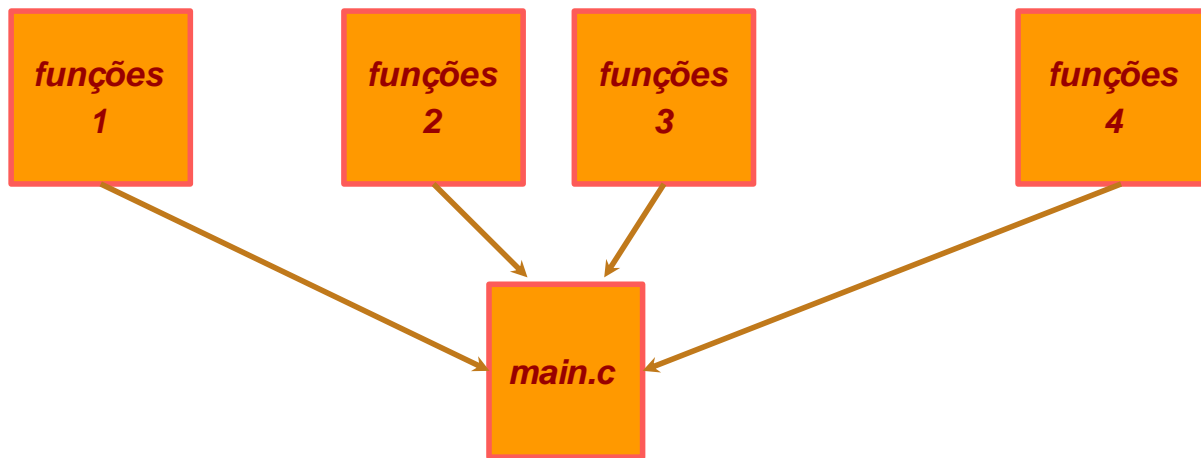
Gerenciando grandes programas em C

- Cada arquivo deve conter uma coleção de códigos relacionados. Por exemplo:
 - Código para leitura e escrita de dados
 - Código para manipulação dos dados



Gerenciando grandes programas em C

- Um arquivo (exemplo: *main.c*) deve chamar as funções definidas em outros arquivos



Gerenciando grandes programas em C

- Já fizemos isso!!!

Chamada aos
arquivos da
biblioteca
padrão C

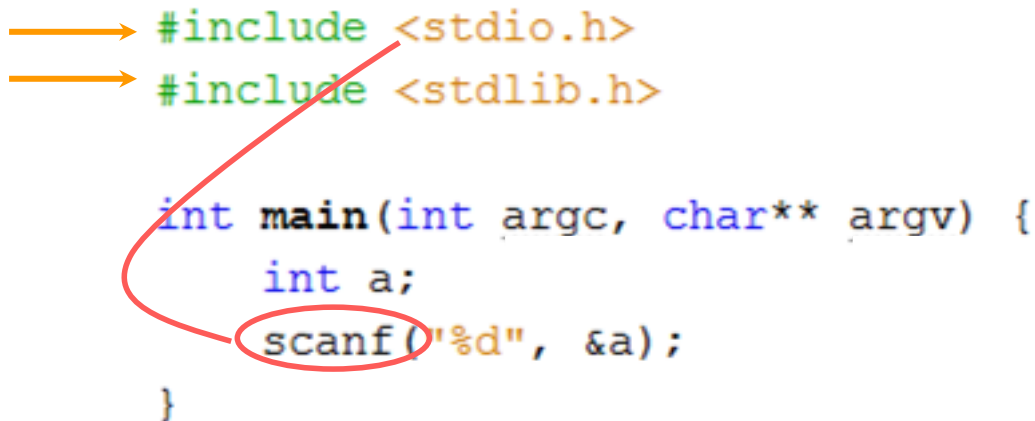
→ `#include <stdio.h>`
→ `#include <stdlib.h>`

```
int main(int argc, char** argv) {  
    int a;  
    scanf("%d", &a);  
}
```


Gerenciando grandes programas em C

- Já fizemos isso!!!

Chamada aos
arquivos da
biblioteca
padrão C



The diagram illustrates the relationship between standard C library functions and their headers. Two orange arrows point from the text 'Chamada aos arquivos da biblioteca padrão C' to the `#include <stdio.h>` and `#include <stdlib.h>` lines. A red circle highlights the `scanf` function call, with a red arrow pointing from the `#include <stdio.h>` line to it, indicating that `scanf` is defined in `stdio.h`.

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

int main(int argc, char** argv) {
    int a;
    scanf("%d", &a);
}
```

Gerenciando grandes programas em C

- Os arquivos de código fonte, com exceção do *main.c*, devem ser divididos em duas partes (ou arquivos):

codigo.h

- Todos os protótipos das funções implementadas em *codigo.c* que não são privadas. Por exemplo, as funções utilizadas pelo *main.c*
- *typedef's*, *enum's* e *struct's*

codigo.c

- Todas as implementações das funções
- Todas as funções privadas (não utilizadas por outros arquivos)

Gerenciando grandes programas em C

- Estrutura de um arquivo **.h** (*header file* - cabeçalho)

```
#ifndef NOME_ARQUIVO
#define NOME_ARQUIVO

//Seus protótipos aqui

#endif
```

#ifndef, **#define** e **#endif** são macros que garantem que o arquivo de cabeçalho será incluído no programa apenas uma vez, mesmo sendo incluído em vários arquivos diferentes do programa.

Gerenciando grandes programas em C

- Exemplo de arquivo *.h*

manipMatrizes.h

```
#ifndef MANIPMATRIZES_H
#define MANIPMATRIZES_H

#include <stdio.h>

#define cols    10

void printMatrix(int m[][cols], int lines);
void sumMatrix(int m[][cols], int n[][cols], int lines);

//...muitas outras funções

#endif
```

Gerenciando grandes programas em C

- Nos arquivos de cabeçalho é possível incluir outros arquivos, da mesma forma que no *main.c*
- Coloque seus arquivos de cabeçalhos no mesmo diretório do arquivo *.c* que implementa os protótipos

Gerenciando grandes programas em C

- Os arquivos de implementação (.c) devem estar no mesmo diretório que os arquivos do seu projeto
- Os arquivos de implementação devem ter a macro **#include** para incluir o cabeçalho que define os protótipos implementados no arquivo .c

Gerenciando grandes programas em C

- Exemplo de arquivo **.c**

manipMatrizes.c

```
#include "manipMatrizes.h"

void printMatrix(int m[][cols], int lines){
    int i,j;
    for(i=0;i<lines;i++){
        for(j=0;j<cols;j++){
            printf("%4d ", m[i][j]);
        }
        printf("\n");
    }
}

void sumMatrix(int m[][cols], int n[][cols], int lines){
    ....
    ....
}
```

Compilando suas bibliotecas

- Há duas formas de compilar um biblioteca:
 - **Estática**
 - Quando a biblioteca for utilizada, ela será incorporada ao seu programa
 - **Vantagem:** Para rodar seu programa, nenhum arquivo será necessário, somente o executável
 - **Desvantagem:** Torna o executável maior

Compilando suas bibliotecas

- Há duas formas de compilar um biblioteca:

- **Compartilhada**

- Quando a biblioteca for utilizada, ela será buscada e utilizada em tempo de execução. Assim, ela não fará parte do programa principal
 - **Vantagem:** Torna o executável menor
 - **Desvantagem:** Todos os arquivos da biblioteca deverão estar na mesma pasta do executável ou em um diretório de `%PATH%`


Compilando suas bibliotecas

- Criando Bibliotecas Estáticas

- No prompt de comando (cmd.exe) , PowerShell ou Terminal:

```
gcc -c arquivos.c -o nomelib.o
```

```
ar rcs libnomelib.a nomelib.o
```



Agrupar as funções em um único
arquivo: *ar - archive*

Compilando suas bibliotecas

- Criando Bibliotecas Estáticas

- No prompt de comando (cmd.exe) , PowerShell ou Terminal:

```
gcc -c arquivos.c -o nomelib.o
```

```
ar rcs libnomelib.a nomelib.o
```



r - replace - substitui se existir

c - create - cria se não existir

s - indexa os arquivos

Compilando suas bibliotecas

- Criando Bibliotecas Estáticas

- No prompt de comando (cmd.exe) , PowerShell ou Terminal:

```
gcc -c arquivos.c -o nomelib.o
```

```
ar rcs libnomelib.a nomelib.o
```



É importante que o arquivo de saída
comece com **lib** e termine com **.a**

Compilando suas bibliotecas

- Criando Bibliotecas Estáticas

- No prompt de comando (cmd.exe) , PowerShell ou Terminal:

```
gcc -c arquivos.c -o nomelib.o  
ar rcs libnomelib.a nomelib.o
```

- Na compilação do programa principal:

```
gcc arquivos_principais.c -L diretorio_lib -l nomelib
```

Perceba que aqui o arquivo não começa com **lib**, nem tem a extensão **.a**



Compilando suas bibliotecas

- Criando Bibliotecas Compartilhadas

- No prompt de comando (cmd.exe) , PowerShell ou Terminal:

```
gcc -c arquivos.c -o nomelib.o
```

```
gcc -shared -o libnomelib.so nomelib.o
```

- Na compilação do programa principal:

```
gcc arquivos_principais.c libnomelib.so
```

Exemplos

Exemplo

Arquivo de cabeçalho com função para calcular a média dos valores de um vetor de tamanho 10.

Exercícios

Exercícios

Você deve fazer um arquivo de cabeçalho *.h* (header file) para cada função; conseqüentemente você terá também um arquivo *.c* (implementação) para cada função. Além disso, você deve fazer o *main.c*.

Se o exercício pedir mais do que uma função, você deve fazer todas separadas em arquivos de cabeçalho *.h* distintos (e também implementar em *.c* distintos).

Exercícios

1. Escreva um arquivo com uma função que receba os comprimentos dos dois lados mais curtos de um triângulo retângulo (catetos) como seus parâmetros. A função deve retornar a hipotenusa do triângulo, calculada usando o teorema de Pitágoras.

Escreva também o programa principal que recebe do usuário os comprimentos dos catetos do triângulo retângulo, chama a função para calcular a hipotenusa e exibe o resultado.