

Fundamentos da Programação

Bibliotecas e Programação Modular

Conteúdo


- Header files
- Biblioteca standard
- Bibliotecas definidas pelo programador

Header file

- Um ficheiro de cabeçalho (header file) é um ficheiro com extensão **.h** que contém normalmente a definição de constantes, estruturas, funções, etc.
 - Exemplos: **stdlib.h**, **stdio.h** e **math.h**.
- Facilita e simplifica o desenvolvimento de aplicações.
 - Permite a separação de elementos de um programa facilitando a sua reutilização noutros programas.
- Inclui-se num programa através da diretiva **#include**, sendo o pré-processador responsável por este processo.

Biblioteca standard

Biblioteca standard

- Contém um conjunto de “funcionalidades”, organizadas por “temas”.
 - A biblioteca **stdio.h** inclui funcionalidades de input/output.
- As bibliotecas incluem:
 - Macros – fragmento de código ao qual é dado um nome, que pode ser utilizado no programa, sendo o nome substituído pelo código no momento da compilação (exemplo, **#define**).
 - Definição de tipos.
 - Funções.
- Exemplo: **#include <math.h>**  entre <...>

math.h

- **double ceil(double x)**
 - retorna o inteiro mais pequeno que seja maior ou igual a **x**.
- **double fabs(double x)**
 - retorna o valor absoluto de **x**.
- **double floor(double x)**
 - retorna o maior inteiro que seja inferior ou igual a **x**.
- **double acos(double x)**
 - retorna o arco cosseno de **x**.
- **double asin(double x)**
 - retorna o arco seno de **x**.

```
#include <stdio.h>
#include <math.h>
int main() {
    int i = 3;
    printf("%d^2 = %lf\n", i, pow(i, 2));
    printf("PI é %lf", M_PI);
    return 0;
}
```

math.h

- **double atan(double x)**
 - retorna o arco tangente de **x**.
- **double cos(double x)**
 - retorna o cosseno de **x**.
- **double sin(double x)**
 - retorna o seno de **x**.
- **double pow(double x, double y)**
 - retorna a potência **x^y**.
- **double sqrt(double x)**
 - retorna a raiz quadrada de **x**.

```
#include <stdio.h>
#include <math.h>
int main() {
    int i = 3;
    printf("%d^2 = %lf\n", i, pow(i, 2));
    printf("PI é %lf", M_PI);
    return 0;
}
```

string.h

- `char *strcat(char *dest, const char *src)`
 - concatena as duas strings.
- `int strcmp(const char *str1, const char *str2)`
 - compara `str1` com `str2`.
- `char *strcpy(char *dest, const char *src)`
 - copia a string `src` para `dest`.
- `size_t strlen(const char *str)`
 - retorna o comprimento da string (sem `'\0'`).

```
#include <stdio.h>
#include <string.h>
#define MAX_STR 50
int main() {
    char uc[MAX_STR] = "Fundamentos de programação";
    printf("\nTamanho da string: %zd", strlen(uc));
    return 0;
}
```



stdlib.h

- **int rand(void)**
 - retorna um valor pseudoaleatório entre 0 to **RAND_MAX**.
- **void srand(unsigned int seed)**
 - inicializar a função rand com um valor "semente" (seed).
- **int atoi(const char *str)**
 - converte uma string para um inteiro.

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int i;
    srand(1);
    for (i = 0; i < 5; i++) {
        printf(" %d ", rand());
    }
    return 0;
}
```

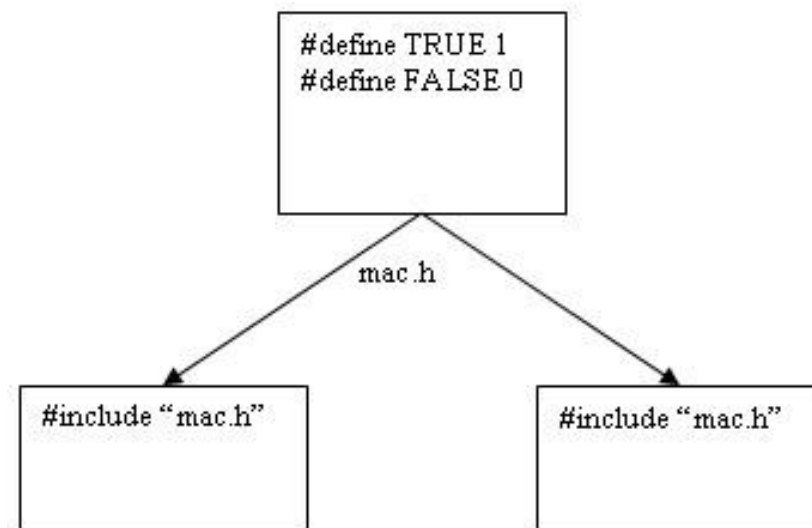
Bibliotecas definidas pelo
programador

Criação de bibliotecas

- O programador pode criar as suas próprias bibliotecas.
- Incluem-se num programa também através da diretiva **#include**, sendo o pré-processor responsável por este processo.
- Exemplo: **#include "nome_ficheiro.h"**  entre "..."

Criação de bibliotecas

- As definições comuns aos dois ficheiros estão armazenadas no header file **mac.h**, quaisquer alterações neste são automaticamente propagadas aos ficheiros que o incluem.



Programação modular

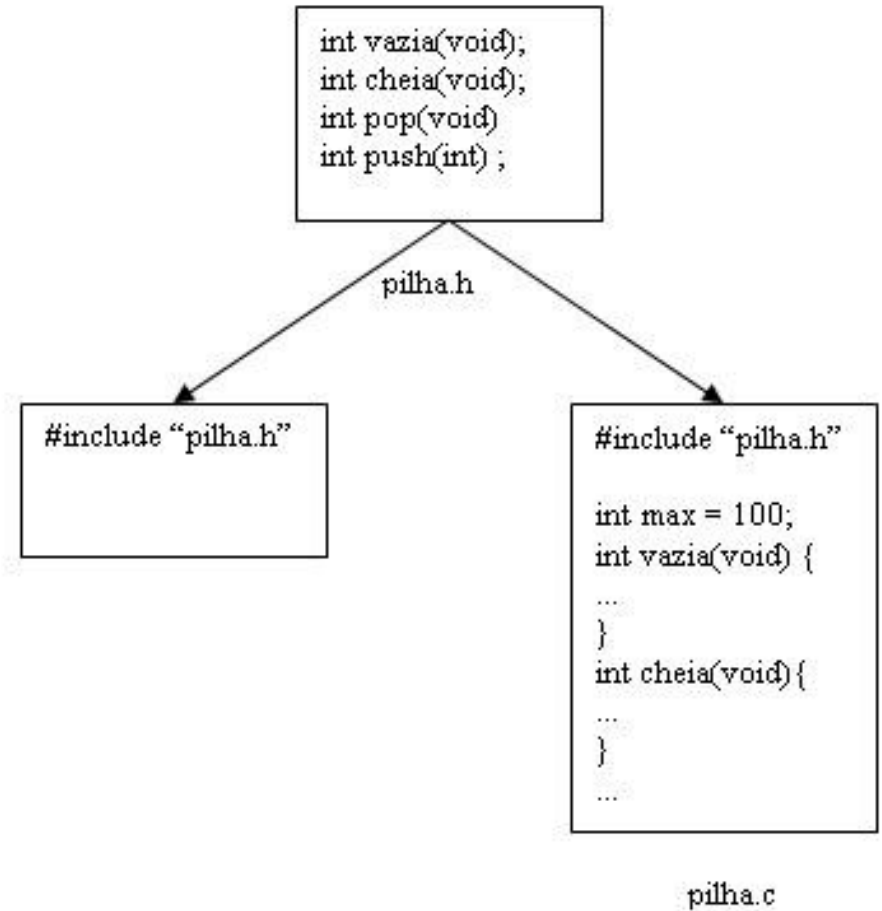
- No desenvolvimento de projetos com alguma dimensão, é usual dividir o código por diversos ficheiros a que se chama **módulos**.
 - Um módulo é uma coleção de funções que realizam tarefas relacionadas.
- Pode-se dividir um módulo em duas partes:
 - Parte pública:
 - Definição de estruturas de dados e funções que devem ser acedidas fora do módulo.
 - Estas definições estão no header file (**.h**) por convenção.
 - Parte privada:
 - Tudo o que é interno ao módulo (não visível pelo mundo exterior).
 - Ficheiro com extensão **.c**.

Programação modular

- A ideia é dividir um problema em secções diferentes e escrever um header file para cada um dos ficheiros C para expor a parte publica (protótipos de funções e constantes, etc.) que podem ser usadas por outros módulos.
- Vantagens:
 - Estrutura do programa fica mais clara.
 - Possibilidade de compilar cada um dos módulos separadamente.
 - A reutilização é facilitada.

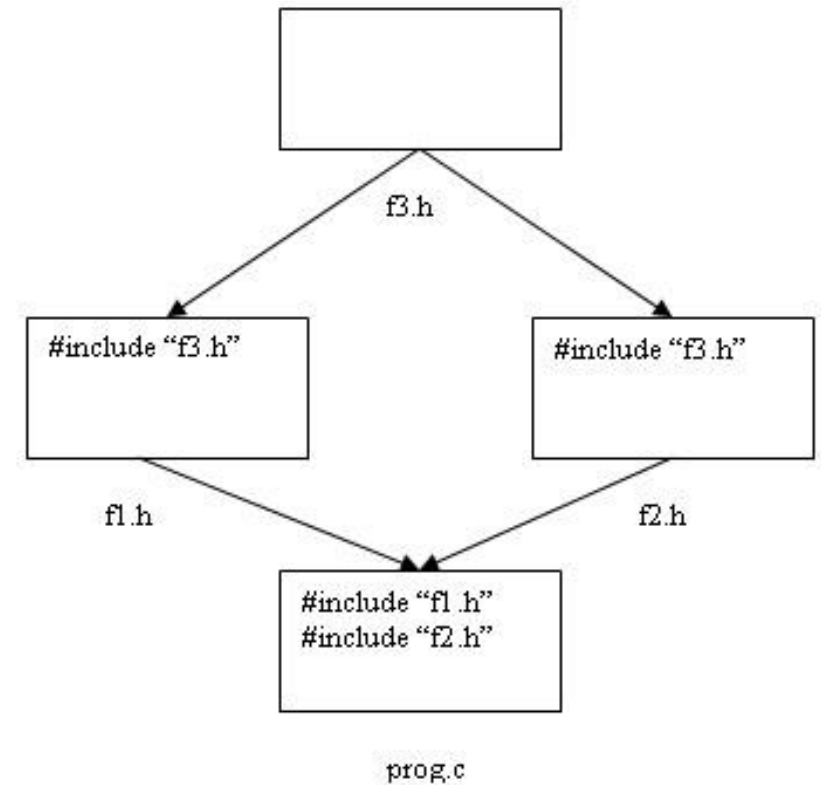
Programação modular

- No caso de um módulo necessitar de aceder a funções definidas noutra, os protótipos respetivos devem estar no header file correspondente.



Programação modular

- A inclusão de um mesmo header file mais do que uma vez origina problemas.
 - Se **f1.h** incluir **f3.h** e **f2.h** incluir **f3.h** e **prog.c** incluir **f1.h** e **f2.h**, **f3.h** será incluído duas vezes o que origina um erro de compilação.



Proteção de header files

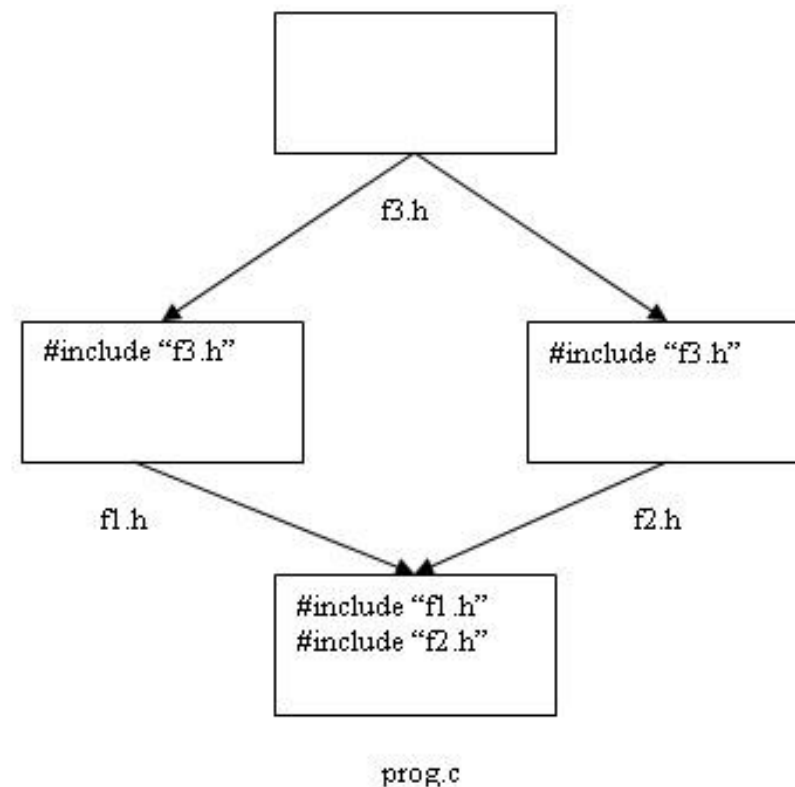
- Para resolver este problema utiliza-se o pré-processador protegendo a inclusão do header file da seguinte forma:

```
#ifndef F3_H
#define F3_H

#define A_STRING "Olá mundo"

void escreveString();

#endif
```



Compilação

- A compilação dos diferentes ficheiros num só executável:

```
$ gcc *.c
```

- ou

```
$ gcc main.c f1.c f2.c f3.c
```

Leitura recomendada

- (Capítulo 13, 14) Damas, L. Linguagem C; FCA – Editora de Informática, Lda, 1999; ISBN 9789727221561.

