ESCOLA SECUNDÁRIA DE SANTO ANDRÉ

CURSO PROFISSIONAL TÉCNICO DE INFORMÁTICA - SISTEMAS

UFCD 0784 - FUNÇÕES

- Programação Modular
- Subprograma
- Conceito de Função
- Exemplos de Funções
- Nomenclatura
- Argumentos de uma função
- Funcionamento de uma Função

- Vantagens das Funções
- Estrutura de uma Função
- Exercícios
- A Instrução Return
- A Função Void
- Variáveis Locais e Globais

ESCOLA SECUNDÁRIA DE SANTO ANDRÉ

CURSO PROFISSIONAL TÉCNICO DE INFORMÁTICA - SISTEMAS

UFCD 0784 - Funções

Objetivos Gerais

- Aprender a estruturar programas;
- Adquirir a noção de subprograma;
- Conhecer as regras de declaração de subprogramas;
- Conhecer as regras de execução de subprogramas;
- Utilizar corretamente argumento;
- Distinguir os diferentes tipos de subprogramas;
- Elaborar programas com recurso a subprogramas;
- Conhecer as regras para a criação de bibliotecas de subprogramas;
- Promover a resolução de exercícios através da programação estruturada.

Conteúdos

- Noção de subprograma;
- Regras de declaração de subprograma;
- Regras de execução de subprograma;
- Argumentos;
- Tipos de subprogramas;
- Programas com recurso a subprogramas;

ESCOLA SECUNDÁRIA DE SANTO ANDRÉ

CURSO PROFISSIONAL TÉCNICO DE INFORMÁTICA - SISTEMAS

AULA 1 (50 min)

Conteúdos Específicos

Tópicos a abordar na aula:

- Revisão dos conceitos:
  - Programação Modular:
    - Conceito
    - Vantagens
- Ciclo de desenvolvimento de um programa em C
- Estrutura de um programa em C
- Exemplos práticos

Sumário da aula 1

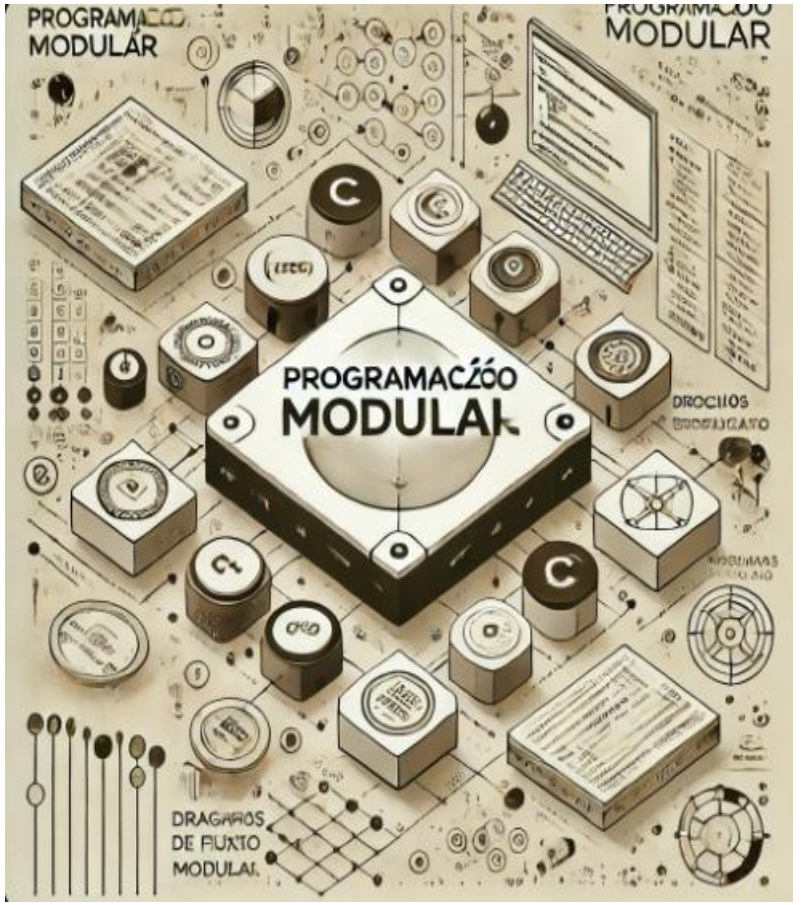
- Introdução ao conceito de Programação Modular e suas principais vantagens.
- Ciclo de desenvolvimento de um programa em C.
- Estrutura de um programa em C.
- A função main().
- Discussão sobre a importância da: função main() e módulos para o desenvolvimento de software.
- Aprendizagem através de exemplos práticos.

PROGRAMAÇÃO MODULAR

- **A programação modular é uma técnica de desenvolvimento de software que envolve a divisão de um programa em partes menores, chamados módulos ou subprogramas.**

- Cada módulo é responsável por realizar uma tarefa específica do programa que pode ser desenvolvido, testado e mantido de forma independente.

- Este conceito é essencial para organizar o código, torná-lo mais legível, facilitar a manutenção e permitir a reutilização de componentes noutros programas.



VANTAGENS

Vantagens da Programação Modular:

- **Divisão de tarefas** – A programação modular divide o programa em várias funções em vez de escrever todo o programa numa única função na main().
- **Reutilização de código** – Os módulos podem ser reutilizados em partes diferentes do programa reduzindo assim a redundância de código.
- **Facilidade na manutenção** – Como o código está dividido torna-se mais fácil encontrar erros e corrigir os mesmos assim como a sua atualização sem modificar todo o código já escrito.

VANTAGENS

Vantagens da programação modular:

- **Legibilidade** – O código fica mais organizado e fácil de compreender porque cada módulo possui uma função bem definida.
- **Facilidade na depuração** – Os problemas encontrados podem ser isolados e resolvidos em módulos individuais, sem influenciar todo o programa.
- **Trabalho de equipa** – Permite que diferentes programadores possam trabalhar em módulos diferentes em simultâneo o que facilita o trabalho em equipa.

A ESTRUTURA DE UM PROGRAMA EM C

Diretivas ao Pré-Processamento

São as instruções processadas antes da compilação

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

Declarações Globais

Variáveis ou funções definidas fora de qualquer subprograma e acessíveis em todas as partes do programa.

```
1   int contador;
2   float calculaArea(float raio);
```

Criação de funções

Funções que organizam tarefas específicas

```
1   float calculaIMC(float peso, float altura)
2   {
3       return peso / (altura * altura);
4   }
```

Função main()

Obrigatória em qualquer programa. Função Principal

```
1   int main()
2   {
3       return 0;
4   }
5
```

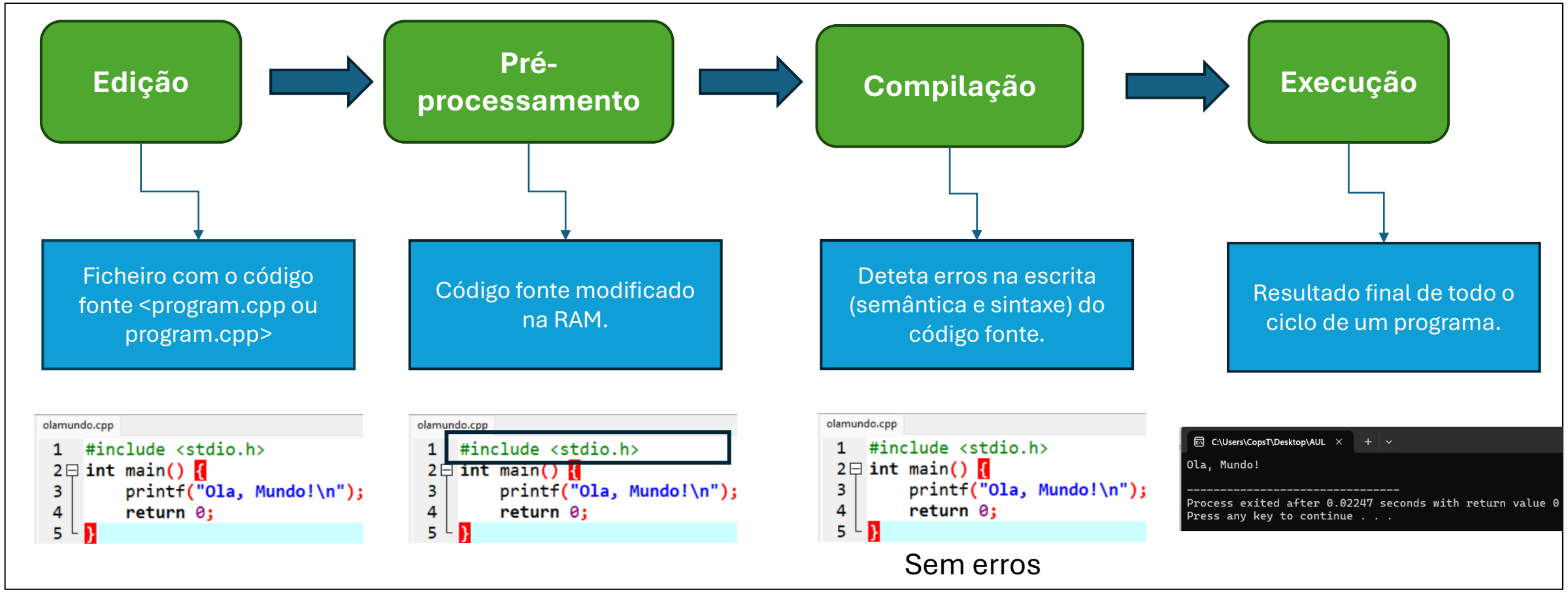
CICLO DE DESENVOLVIMENTO DE UM PROGRAMA

- **É um processo que transforma o código-fonte escrito pelo programador num programa executável.**

- É composto por várias etapas:
  - edição;
  - pré-processamento;
  - compilação;
  - execução.

- Tem como objetivo garantir que o programa funciona corretamente e sem erros.

CICLO DE DESENVOLVIMENTO DE UM PROGRAMA



EDIÇÃO

- Momento principal e fundamental em que o programador escreve o código num editor de texto ou IDE como por exemplo o Dev C++.
- É das etapas fundamentais em que o programador além de escrever o código cria o programa com uma determinada lógica e com as funcionalidades desejadas.
- O ficheiro criado deve conter sempre nome de program.cpp

olamundo.cpp

```
1  #include <stdio.h>
2  int main() {
3      printf("Ola, Mundo!\n");
4      return 0;
5  }
```

PRÉ-PROCESSAMENTO

- O pré-processador trabalha o código de forma a que fique preparado para ser compilado.
- Sem este procedimento do #include, o código não reconheceria a função printf().
- O pré-processador em C realiza uma série de transformações no código fonte antes que o compilador comece a compilar o programa.

olamundo.cpp

```
1  #include <stdio.h>
2  int main() {
3      printf("Ola, Mundo!\n");
4      return 0;
5  }
```

COMPILAÇÃO

- **Tradutor principal do código fonte para a linguagem máquina.**
- **Identifica erros de sintaxe e de semântica no código fonte.**
- Cria ficheiros intermediários para a próxima etapa.
- O código da máquina é uma sequência de instruções que o processador do computador executa.
- Funciona como um conversor uma vez que lê o código em C e converte-o para código de máquina.

olamundo.cpp

```
1  #include <stdio.h>
2  int main() {
3      printf("Ola, Mundo!\n");
4      return 0;
5  }
```

Sem erros

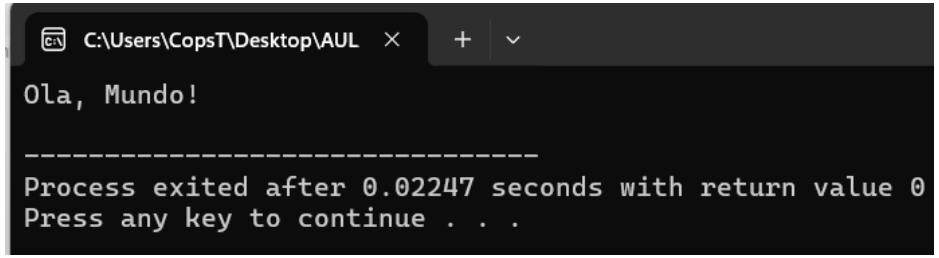
olamundo.cpp

```
1  #include <stdio.h>
2  int main() {
3      printf("Ola, Mundo!\n");
4      return 0};
5
```

Com erros

EXECUÇÃO

- **O programa executável (program.exe) está pronto nesta fase para ser executado.**
- **Testar o programa e verificar se funciona como esperado. Após cada etapa podem existir erros e é necessário corrigir.**
- O utilizador interage com o programa e valida a saída.
- Durante as etapas da edição, pré-processamento e compilação o programador pode corrigir erros encontrados.



A FUNÇÃO MAIN()

- **A função main() é o ponto de entrada de qualquer programa em C**.
- **Estrutura básica do C:**

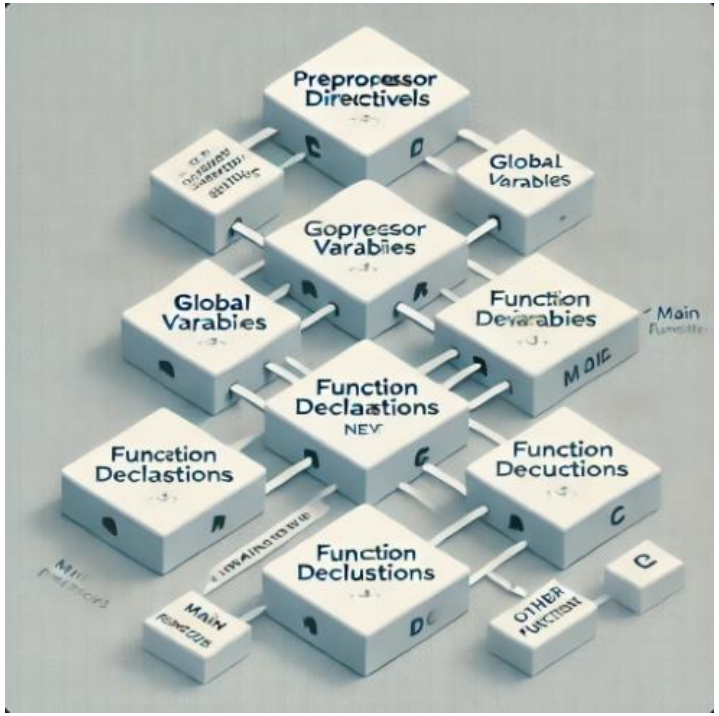
```
#include <stdio.h>
int main()
{
    printf("Olá, Mundo!\n");
    return 0;
}
```

- Inicio e término na função main().
- O valor de retorno (return 0) indica que o programa terminou com sucesso.

ESTRUTURA DE UM PROGRAMA EM C

```
/* Primeiro Programa em C */ comentários

#include <stdio.h> /*biblioteca de E/S */
 SUBPROGRAMAS
main()      /*função principal - inicio do
             programa*/
{            /*marca início da função*/
   printf("Meu primeiro programa em C\n");
            /*escreve no monitor e muda de
  linha*/
}            /*marca o fim da função*/
```



EXEMPLO DE UM PROGRAMA EM C

Neste exemplo, temos um programa que utiliza funções separadas para somar e subtrair dois números.

- Quantas funções se encontram neste exemplo?
- Existe alguma função da própria biblioteca de funções do C? Se sim qual?
- Quantas funções neste exemplo foram criadas?

[*] CODIGO.cpp

```
1  #include <stdio.h>
2  int a, b;
3  int soma(int a, int b)
4  {
5      return a + b;
6  }
7  int subtrai(int a, int b)
8  {
9
10     return a - b;
11 }
12 int main()
13 {
14     //int num1 = 10, num2 = 5;
15
16     printf("Soma: %d\n", soma(a, b));
17     printf("Subtracao: %d\n", subtrai(a, a));
18
19     return 0;
20 }
```

RESUMO DA AULA

• Falámos no conceito de Programação Modular e suas vantagens.
• Falámos sobre o Ciclo de desenvolvimento de um Programa e suas etapas.
• Vimos como o código-fonte pode ser transformado num programa executável.
• Analisámos a Estrutura de um Programa em C.
• Introdução à função main().
• Vimos exemplos de estruturas e código simples.
• Realizámos exemplos práticos através de demonstrações simples para exemplificar os conceitos abordados.

PRÓXIMA AULA

- Iremos falar no conceito de Função vs Subprograma.
- Tipos de Subprogramas.
- Realizar um review sobre as três principais funções: main(); Printf() e scanf().
- Perceber a Nomenclatura das funções.
- Características de uma função.
- Ver Parâmetros e argumentos de uma função.
- Funcionamento de uma função e as suas vantagens.
- Funções com instruções e Estrutura de uma função.
- Exercícios conduzidos step by step.

FIM DA AULA 1