

## Aula 1

### Linguagem de Programação

Prof. Sandro de Araújo

### Conversa Inicial

### Plano de ensino

- Aula 1 – Estrutura básica de um programa em C
- Aula 2 – Ponteiros
- Aula 3 – Registros: *struct*, *union*, *enum* e *typedef*
- Aula 4 – Ponteiros: *struct*, função, alocação dinâmica
- Aula 5 – Recursividade e macros
- Aula 6 – Arquivos e operações com *bits*

### Temas desta aula

1. Compiladores
2. Estrutura de um programa em C
3. Função *main*
4. O nome das funções
5. Pré-processador e diretivas

### Compiladores

- O nome *compilador* se refere ao processo de tradução
- Texto-fonte – código-fonte → homem
- Texto-objeto – código-objeto → máquina



- A tradução de uma linguagem-fonte não é a única função do compilador
- Ele também reporta ao seu usuário a presença de erros no programa de origem

- Existem duas tarefas principais executadas por um compilador no processo de tradução:
  - Análise – o momento em que o texto de entrada (código-fonte) é examinado, verificado e compreendido
  - Síntese (ou geração de código) – momento em que o texto de saída (código-objeto) é gerado

- Segundo Mizrahi (2008), alguns compiladores costumam dividir o processo de tradução em várias etapas e executá-los em sequência para um melhor aproveitamento da memória durante a execução

- Cada etapa constitui uma parte do processo de tradução, transformando, assim, o código-fonte em estrutura intermediária adequada mais próxima do código-objeto final



**Arquivo olaMundo.c**

```

1. #include <stdio.h>
2. void main()
3. {
4.     printf("Olá Mundo!");
5. }
Programa fonte
  
```

**Pré-processado**

**Compilado  
olaMundo.s**  
Representação  
intermediária

**Vinculado (Linker)  
olaMundo.exe**  
Programa executável

**Montado  
olaMundo.o**  
Programa objeto

**Estrutura de um programa em C**

- A linguagem de programação C é uma linguagem de alto nível, com sintaxe estruturada e flexível
- Com essa linguagem criamos programas compilados, gerando programas executáveis

- Um programa em C é constituído de:
  - Cabeçalho – inclusão de bibliotecas, diretivas de compilador nas quais se define o valor de constantes simbólicas, declaração de variáveis, declaração de funções, entre outros
  - Bloco principal – de instruções e outros blocos de rotinas
  - Documentação do programa – em forma de comentários

- Os comentários podem ser escritos em qualquer parte do algoritmo, para que o comentário seja identificado como tal
- O comentário deve ter um /\* antes e um \*/ depois para comentar um bloco, ou // para comentar apenas uma linha

```
1. #include <stdio.h>
2. void main()
3. {
4.     printf("Olá Mundo!");
5. }
```

- A primeira linha do programa #include <stdio.h> informa ao compilador qual biblioteca deverá incluir a biblioteca stdio (standard input/output) no programa

- Na segunda linha foi declarada a única função, a função main, e nesta existe apenas uma única instrução:
  - função printf() (disponível na biblioteca stdio.h da linguagem C) para escrever uma mensagem no monitor

## Função *main*

- A função *main* serve como ponto de partida para a execução do programa
- Em geral, ela controla a execução direcionando as chamadas para outras funções no programa

## Formatos usados na função *main*

- `int main()`
- `int main(void)`
- `int main(int argc, char * argv[ ])`
- `int main(int argc, char * const argv[ ], char * const envp[ ])`

## `int argc`

- `argc` – contador de argumentos; informa quantos argumentos foram passados juntos com o nome do programa

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(int argc, char * const argv[ ], char * const envp[ ])
5 {
6     printf("O conteudo de argc = %d\n", argc);
7
8     return 0;
9 }
10
```

```
"C:\Users\Casa\Documents\Sandro\FACULDADES\UNINTER\Linguagem de Programação\fu
0 conteudo de argc = 1
Process returned 0 (0x0)   execution time : 0.334 s
Press any key to continue.
```

## `argv[]`

- Vetor de ponteiros, em que cada ponteiro indica um argumento passado; o nome do programa é armazenado em `argv[0]`

```
"C:\Users\Casa\Documents\Sandro\FACULDADES\UNINTER\Linguagem de Programação\fu
conteudo de argc = 1
Process returned 0 (0x0)   execution time : 0.334 s
Press any key to continue.
```

```

1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(int argc, char * const argv[ ], char * const envp[ ])
5  {
6      printf("O conteudo de argv = %s\n", argv[0]);
7      return 0;
8  }
9

```

## envp[]

- Um ponteiro para um vetor de *strings*, com informações sobre o ambiente do processo

```

C:\Users\Casa\Documents\Sandro\FACULDADES\UNINTER\Linguagem de Programação\funcãoMain\bin\Debug\funcaoMain.exe
O conteudo de envp na posicao 0 = ALLUSERSPROFILE=C:\ProgramData
O conteudo de envp na posicao 1 = APPDATA=C:\Users\Casa\AppData\Roaming
O conteudo de envp na posicao 2 = CommonProgramFiles=C:\Program Files (x86)\Common Files
O conteudo de envp na posicao 3 = CommonProgramFiles(x86)=C:\Program Files (x86)\Common Files
Process returned 0 (0x0)   execution time : 0.260 s
Press any key to continue.

```

```

1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(int argc, char * const argv[ ], char * const envp[ ])
5  {
6      printf("O conteudo de envp na posicao 0 = %s\n", envp[0]);
7      printf("O conteudo de envp na posicao 1 = %s\n", envp[1]);
8      printf("O conteudo de envp na posicao 2 = %s\n", envp[2]);
9      printf("O conteudo de envp na posicao 3 = %s\n", envp[3]);
10     return 0;
11 }

```

## Principais características de uma função

### Formatos usados na função *main*

- Toda função é declarada com uma identificação e parênteses após seu nome. Por exemplo: `main()`
- Sem os parênteses, o compilador pode tratar o nome como se fosse uma variável

- Toda função em linguagem de programação C delimita o bloco com chaves

```

int man(){

}

```

- Quando desenvolvemos um projeto de criação de *software*, é importante que todo o código seja bem indentado, ou seja, alinhado de forma correta

- Num projeto grande, é importante a definição de um padrão de indentação
- Esse padrão deve ser documentado e disponibilizado para os demais programadores

- A palavra *indentação* é um neologismo e não existe na língua portuguesa
- Foi “abrasileirada” do termo *indentation*, usado na língua inglesa, que significa recuo

```
int soma(int a, int b);

int main ()
{
    soma(5, 3);
    return 0;
}

int soma(int a, int b)
{
    int r;
    r = a + b;
    printf("O resultado e': %d", r);
}
```

### Principais características de uma função

- As primeiras linhas de um programa não são instruções da linguagem C (observe que não há ponto e vírgula no final), mas sim diretivas do pré-processador
- Exemplo:
  1. `#include <stdio.h>`
  2. `#include <stdlib.h>`

- Toda diretiva é iniciada pelo símbolo (#), um código especial
- Seu texto deve ser escrito em uma única linha

### Códigos especiais

- Além do comando #, existem vários outros caracteres usados para auxiliar o programador em seu código
- A barra invertida (\) é um comando usado no momento em que o programador precisa digitar algo que não pode ser digitado diretamente no teclado

CÓDIGOS ESPECIAIS	SIGNIFICADO
\n	Nova linha
\t	Tabulação
\f	Salto de página de formulário
\r	CR – Retorno do cursor p/ o início da linha
\\	Barra invertida
\0	Zero
\'	Aspas simples (apóstrofo)
\"	Aspas duplas

- Portanto, na linguagem de programação C, sempre que aparecer uma barra invertida ('\') ou um símbolo de porcentagem ('%'), será um comando

Códigos de impressão formatada com printf()	SIGNIFICADO
%c	Caractere simples
%d	Inteiro decimal com sinal
%i	Inteiro decimal com sinal
%e	Notação científica (e minúsculo)
%E	Notação científica (e maiúsculo)
%f	Ponto flutuante em decimal