

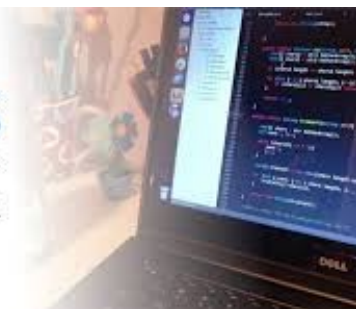
# *Tecnologia em Análise e Desenvolvimento de Sistemas - TADS*

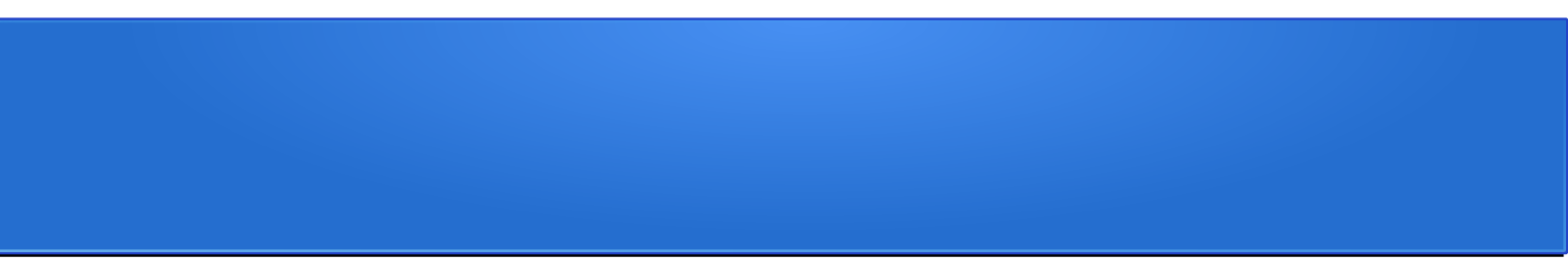
## ***Estrutura de Dados I***

*Atividades Pedagógicas Não Presenciais – APNP 2020*

***Prof. Luciano Vargas Gonçalves***

*E-mail: [luciano.goncalves@riogrande.ifrs.edu.br](mailto:luciano.goncalves@riogrande.ifrs.edu.br)*





# Aula 1 – Revisão da Linguagem C

# Olá Mundo em C

```
/* Primeiro Programa em C */  
#include <stdio.h>  
  
main()  
{  
printf("Meu primeiro programa em C\n");  
}
```

# Olá Mundo em C

## Programa comentado

```
/* Primeiro Programa em C */ comentários
#include <stdio.h> /* biblioteca de E/S */

main() /* função principal - início do programa */
{
    /* marca início da função */
    printf("Meu primeiro programa em C\n");
    /* função para escrever na tela */
} /* marca fim da função */
```

Saída na tela Comando – `printf(".....");`

# Olá Mundo em C

**Digite o código em um editor de texto. Exemplo “CodeBlocks”**

```
/* Primeiro Programa em C */  
#include <stdio.h>  
  
main()  
{  
printf("Meu primeiro programa em C\n");  
}
```

# Linguagem C

- Uma Variável: que pode assumir diversos valores;
  - Espaço de memória de um certo tipo de dado associado, possui um nome para referenciar seu conteúdo.
  - Tipagem forte, necessita declaração

```
{  
    int idade;  
    idade = 30;  
    printf (" A idade é : %d", idade);  
}
```

# Linguagem C

- Nome de Variáveis

- Quantos caracteres quiser (32);
- Começar com letras ou sublinhado:
  - Seguidos de letras, números ou sublinhados
- C é sensível ao caso:

peso    <>    Peso    <>    pEso

- não podemos definir um identificador com o mesmo nome que uma palavra-chave, Exemplos:
  - auto static extern int long if while do .....

# Linguagem C

## Declaração de Variáveis

- Instrução para reservar uma quantidade de memória para um certo tipo de dado, possui um nome para ser referenciada dentro do programa;
  - Definição
    - tipo nome-da-variável;
    - tipo nome1, nome2,..., nome;
      - EX:
        - char nome;
        - int idade, num = 10;



# Linguagem C

## Declaração de Variáveis – Tipos primitivos

tipo	bytes	escala
char	1	-128 a 127
int	2	-32.768 a 32.767
float	4	3.4e-38 a 3.4e+38
double	8	1.7e-308 a 1.7e+308

<b>Long ou Long int</b>	<b>(4 bytes)</b>
<b>Unsigned Char</b>	<b>(0 a 255)</b>
<b>Unsigned int</b>	<b>(0 a 65.535)</b>

# Linguagem C

## Declaração de Variáveis – Tipos primitivos

```
#include <stdio.h>
main( )
{
    int soma=10;
    float money=2.21;
    char letra= 'A';
    double pi=2.01E6;

    printf ("valor da soma = %d\n", soma);
    printf ("Valor de Money = %f\n", money);
    printf("Valor de Letra = %c\n", letra);
    printf("Valor de Pi = %e\n", pi);

}
```

# Entrada de Dados

- Leitura de dados tipados via teclado
  - *Scanf* (“string de controle”, lista de argumentos);

Exemplo:

```
scanf("%d",&idade);
```

**OBS**: Para sequência de caracteres (%s), o caracter **&** não deverá ser usado.

# Saída de Dados - Printf

- Apresentação de dados no monitor
- `printf("string de controle", lista de argumentos);`

Exemplo:

```
printf ("Digite a sua idade:\n");
```

```
scanf ("%d", &idade);
```

```
printf("Sua idade é: %d", idade);
```

# Entrada e Saída de Dados – Caracteres de formatação

## Caracteres especiais e de formatação texto

- **%c** → **caracter**
- **%d** → **inteiro**
- **%e** → **número ou notação científica**
- **%f** → **ponto flutuante**
- **%o** → **octal**
- **%X** → **hexadecimal**
- **%s** → **string (cadeia de caracteres)**
- **%lf** → **double**

# Entrada e Saída de Dados

## Exemplos de usos dos caracteres especiais

```
#include <stdio.h>
main ( )
{
    char a ;
    printf ( “digite um caracter” );
    scanf ( “ % c”, &a );
    printf ( “ \n %c = %d em decimal”, a, a);
    printf (“%o em octal, %x em hexadecimal”, a, a);
}
```

**Digitando m:**

m = 109 em decimal, 155 em octal, 6d em hexadecimal

# Operador de endereço “&”

- Um endereço de memória é o nome que o computador usa para identificar uma variável ou espaço da memória;
- Toda variável ocupa uma área de memória e seu endereço é o primeiro byte por ela ocupado

Ex :

inteiro → 2 bytes

float → 4 bytes

char → 1 byte

Espaço ocupado por um variável deste tipo

# Operador de endereço "&"

- Quando usamos & precedendo uma variável estamos falando do endereço desta variável na memória

Ex:

```
Main ( )
```

```
{
```

```
    int num;
```

```
    num = 2;
```

```
    printf ("valor = %d, endereço = %Iu", num, &num);
```

```
}
```

Saída: valor = 2, endereço = 1230

Varia conforme memória da máquina

Endereço	Valor
1230	2
1231	
1232	
1233	

Ocupa dois bytes de memória



# Saída de Dados

- Comando printf formatação para REAIS

```
#include <stdio.h>
main ( )
{
    printf ("os alunos são %2d \n", 350);
    printf ("os alunos são %4d \n", 350);
    printf ("os alunos são %5d \n", 350);
}
```

Saída:    os alunos são 350  
         os alunos são 350  
         os alunos são    350

Ocupa 5 posições

# Saída de Dados

- Comando printf formatação para REAIS

```
#include <stdio.h>
main ( )
{
    printf ("%3.1f \n", 3456.78);
    printf ("%10.3f \n", 3456.78);
}
```

Saída:      3456.8

          3456.780

Ocupa 3 posições após a vírgula

# Exemplo – Array de Char ( String)

Programa para ler o nome e a idade de uma pessoa, Após mostrar na tela os valores:

```
#include <stdio.h>
main( )
{
    int idade;
    char nome[30];
    printf ("Digite o seu nome:\n");
    scanf("%s", nome);
    printf ("Digite a sua idade:\n");
    scanf ("%d", &idade);
    printf("A idade do(a) %s é %d", nome, idade);
}
```

Vetor não usa "&"



# Operadores Aritméticos

Operador	Ação
+	Adição
*	Multiplicação
/	Divisão
%	Resto da divisão inteira
-	Subtração (unário)
--	Decremento
++	Incremento

# Operadores relacionais e lógicos

Operador	Ação
>	Maior que
>=	Maior ou igual que
<	Menor que
<=	Menor ou igual que
==	Igual a
!=	Diferente de
&&	Condição "E"
	Condição "OU"
!	Não

# Operadores relacionais e lógicos

- Em C o resultado da comparação será ZERO se resultar em FALSO
- DIFERENTE DE ZERO no caso de obtermos VERDADEIRO num teste qualquer.
- $12 > 15$  “saída “0” que representa falso”
- $12 < 15$  “saída “1” que representa verdadeiro”

# Operadores relacionais e lógicos

## Exemplo

```
#include <stdio.h>
main ( )
{
    int verdadeiro, falso;

    verdadeiro = (15 < 20);
    falso      = (15 == 20);
    printf("Verd.= %d,Falso= %d",
        verdadeiro, falso);
}
```

Saída: Verd.=1 Falso = 0

# Comparação

- Observemos antes de mais nada que  $++x$  é diferente de  $x++$ !

Se

```
x = 10;
```

```
y = ++x;
```

```
/* x=x+1; y=x; */
```

então

```
x = 11 e
```

```
y = 11
```

• Pré – incremento

- Incrementa X e atribui

porém Se

```
x = 10;
```

```
y = x++;
```

```
/* y=x; x=x+1 */
```

então

```
x = 11 e
```

```
y = 10
```

Pós incremento

Atribui e depois incrementa



# Comparação

```
if (10 > 4 && !(10 < 9) || 3 <= 4)
```

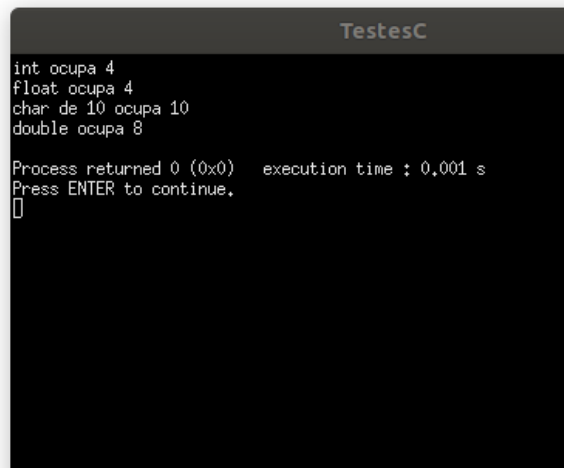
**Como seria avaliado esta instrução?**

resultaria em Verdadeiro, pois dez é maior que quatro E dez não é menor que nove OU três é menor ou igual a quatro

# Operador SIZEOF

- Este operador retorna o tamanho da variável ou tipo que está em seu operando.

```
4  int main()  
5  {  
6      int a = 10;  
7      float b = 19.5;  
8      char c [10] = "ola mundo";  
9      double d = 25.8;  
10  
11  
12      printf("int ocupa %d\n", sizeof(a));  
13      printf("float ocupa %d\n", sizeof(b));  
14      printf("char de 10 ocupa %d\n", sizeof(c));  
15      printf("double ocupa %d\n", sizeof(d));  
16      return 0;  
17  }
```



```
TestesC  
int ocupa 4  
float ocupa 4  
char de 10 ocupa 10  
double ocupa 8  
  
Process returned 0 (0x0)   execution time : 0,001 s  
Press ENTER to continue.  
█
```

# Operadores de Seleção - IF

- Permitir testes para decidir ações alternativas:
  - if
  - if – else
  - switch
  - (?:) Operador Condicional

# Operadores de Seleção - IF

- Comando getchar() captura um character;

```
#include <stdio.h>
main ( )
{
    char ch;
    ch = getchar ( );
    if (ch == 'p')
        printf ("você pressionou a
tecla p");
}
```

```
#include <stdio.h>
main ( )
{
    if (getchar() == 'p' ) {
        printf (" você digitou p");
        printf (" pressione outra tecla ");
        getchar( );
    }
}
```

# Operadores de Seleção – Ternário ?

- Operador Ternário ?
  - Forma compacta de expressar uma instrução if – eles
    - (condição) ? expressão1 : expressão2
  - Exemplo:
    - `Max = (num1 > num2) ? num1 : num2`
    - Equivalente:
      - `if (num1 > num2)    max = num1;   else max = num2;`
    - Exemplo com atribuição
      - `ABS = (num < 0) ? - num : num;`

# Operadores de Seleção – Switch/Case

```
switch <variável> {  
    case <constante 1> :  
        <comandos>;  
        [break;]  
    case <constante 2> :  
        <comandos>;  
        [break;]  
    case <constante 3> :  
        <comandos>;  
        [break;]  
    [default :  
        <comandos>;]  
}
```

## OBS:

- “variável” deve ser uma variável do tipo inteiro ou caracter;
- “break” serve para terminar a seqüência de comandos em execução, por serem opcionais, se forem suprimidos permitem que o “case” a seguir seja executado, sem haver qualquer quebra na seqüência do processamento.

# Estrutura de Repetição - For

***for (<início>;<condição>;<incremento>) <comando>;***

Na forma mais simples:

- Inicialização:
  - expressão de atribuição
  - sempre executada uma única vez
- Teste:
  - condição que controla a execução do laço
  - é sempre avaliada a cada execução
  - verdadeiro → continua a execução
  - falso → para a execução

# Estrutura de Repetição - For

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

```
main ( )
```

```
{
```

```
    int número;
```

```
    for ( número = 2; número < 10; número += 2 )  
        printf ( " %d", número );
```

```
}
```

**Saída 2 4 6 8**

## Exemplos

```
for (x=0,y=0;x+y<100;++x,y=y+x)  
    printf("%d",x+y);
```

Exemplo com duas variáveis de condições



# Estrutura de Repetição – For - LOOP

```
for(;;)  
    printf("Este loop rodará eternamente!\n");
```

## Exemplos

```
for (i=0 ; i<10 ; i++) ;
```

- A presença do ponto e vírgula finalizando o comando, força a execução do loop sem que seja executado qualquer outro comando.

# Estrutura de Repetição – While

```
while <condição> <comando>;
```

Exemplo: Contagem

```
#include <stdio.h>
main()
{
    int i=0;
    while (i < 10) {
        printf("%d", i);
        i++;
    }
}
```

- O loop se repete, enquanto a condição for verdadeira

# Estrutura de Repetição – do / while

- Ao contrário das estruturas “for” e “while” que testam a condição no começo do loop, “do / while” sempre a testa no final, garantido a execução ao menos uma vez da estrutura.

```
do {  
    <comandos>;  
} while <condição>;
```

Exemplo: Término determinado pelo usuário.

```
#include <stdio.h>  
main()  
{  
    int num;  
    do {  
        scanf(“%d”,&num);  
    } while (num < 100);  
}
```

- 
- **Próxima Aula:**
    - **Ponteiros e Structs**