

# LÓGICA DE PROGRAMAÇÃO

PROF<sup>a</sup>. M.Sc. JULIANA H Q BENACCHIO

# Primeiro programa em C

```
#include <stdio.h>

int main()
{
    int num1, num2, result;

    scanf ("%d", &num1);
    scanf ("%d", &num2);

    result = num1 + num2;
    printf ("%d", result);

    return 0;
}
```

# Primeiro programa em C

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

```
int main()
```


```
{
```

```
    int num1, num2, result;  Declaração de variáveis
```

```
    scanf ("%d", &num1);
```

```
 Entrada
```

```
    scanf ("%d", &num2);
```

```
    result = num1 + num2;  Processamento
```

```
    printf ("%d", result);  Saída
```

```
    return 0;
```

```
}
```

# Diretivas `#include`

*`#include <nome_do_arquivo>`*

- Arquivos que contêm bibliotecas de funções ou rotinas do próprio C ou do usuário.
  - `stdio.h` - Funções de entrada e saída
  - `stdlib.h` - Funções de uso genérico
  - `math.h` - Funções matemáticas
  - `ctype.h` - Funções para caracteres

# Tipos

- **void** – sem valor
- **char** – um único caractere
- **int** – números inteiros
- **float** – números em ponto flutuante com precisão simples (7 casas decimais)

# Declaração de Variáveis

```
tipo_da_variavel lista_de_variaveis;
```

- Devem ser declaradas antes de serem usadas

```
char ch, letra;
```

```
int num;
```

```
float pi;
```

# Inicialização de Variáveis

```
tipo_da_variavel nome_da_variavel = constante;
```

- quando o C cria uma variável ele não a inicializa
- Nunca presume que uma variável declarada vale zero ou qualquer outro valor

# Declaração de Variáveis

- Deve ser declarada no início de um bloco de código

```
int main ()  
{  
  
    int i, j; //Variaveis locais  
  
    printf ("Ola Mundo!\n");  
  
    return 0;  
  
}
```



# Operador de atribuição

```
variavel = valor;
```

- Inicialização de variáveis

```
int num = 10;
```

```
float pi = 3.14;
```

# Entrada e Saída

---

- `scanf ()` - leitura de dados (tipados) via teclado
- `printf ()` - apresentação de dados (formatados) na tela

# Leitura: scanf ( )

```
scanf ("expr_de_controle", lista_de_arg) ;
```

- A expressão de controle deve conter a formatação do tipo de variável a ser lida
  - %c - leitura de caractere
  - %d - leitura de números inteiros
  - %f - leitura de números reais

# Leitura: scanf ( )

```
scanf ("expr_de_controle", lista_de_arg) ;
```

- A lista de argumentos deve constar apenas endereço de variáveis
  - `scanf ("%d", &num) ;`
- Pode-se utilizar mais de uma variável na lista de argumentos
  - `scanf ("%d %d", &num1, &num2) ;`

# Escrita: `printf()`

```
printf("expr_de_controle", lista_de_arg) ;
```

- Necessariamente você precisará ter tantos argumentos quantos forem os comandos de formatação na expressão de controle

# Escrita: `printf()`

```
printf("expr_de_controle", lista_de_arg);
```

- Os caracteres a serem utilizados pelo `printf()` em sua `expr` de controle serão os mesmos de `scanf()`
  - `%c` – imprime um caractere
  - `%d` – imprime um número inteiro
  - `%f` – imprime um número real
  - `%%` - imprime o símbolo `%`

# Exemplo - scanf () e printf ()

```
int main()
{
    int num, dobro;
    printf("Digite um numero: ");
    scanf("%d", &num);
    dobro = num * 2;
    printf("\0 dobro de %d = %d", num, dobro);
    return 0;
}
```

# Operadores

- Operadores aritméticos
- Operadores de incremento e decremento
- Operadores aritméticos de atribuições
- Operadores relacionais
- Operadores lógicos
- Operadores bit a bit
- Operador **cast**



# Operadores aritméticos

- Os operadores aritméticos são usados para calcular expressões matemáticas com os seus operando
- Estes operandos podem ser utilizados com qualquer tipo de dados, exceto o resto da divisão, o qual não pode ter operandos em ponto flutuante
- Os operadores aritméticos são classificados em duas categorias:
  - Binários – 2 operandos
  - Unários – 1 operando

# Operadores aritméticos

- Operadores aritméticos binários:
  - $+$  → Soma
  - $-$  → Subtração
  - $*$  → Multiplicação
  - $/$  → Divisão
  - $\%$  → Resto da divisão

# Operadores aritméticos

- Os operadores aritméticos unários atuam na inversão de valores
  - - → Sinal negativo
  - + → Sinal positivo

# Operadores de Incremento e Decremento

- O operador de incremento (`++`) soma 1 ao seu operando . Por exemplo, `c++` pode ser usado no lugar da expressão `c = c + 1`
- De forma análoga, o operador de decremento (`--`) subtrai 1 do seu operando.

# Operadores aritméticos de atribuições

- São combinações de operadores que simplificam as instruções. Dessa forma uma instrução escrita da forma:

$$x = x \text{ op } y;$$

- Pode ser reduzida obedecendo à sintaxe:

$$x \text{ op} = y;$$

# Operadores aritméticos de atribuições

## • Expressão Normal

**a = a + b;** →

**a = a - b;** →

**a = a \* b;** →

**a = a / b;** →

**a = a % b;** →

## Expressão Simplificada

**a+= b;**

**a-= b;**

**a\*= b;**

**a/= b;**

**a%= b;**

# Operadores relacionais

- Os operadores relacionais são utilizados em expressões condicionais para a comparação do valor de duas expressões:
  - $>$  → Maior que
  - $>=$  → Maior ou igual à
  - $<$  → Menor que
  - $<=$  → Menor ou igual à
  - $==$  → Igual à
  - $!=$  → Diferente de

# Operadores lógicos

- Os operadores lógicos são utilizados para conectar expressões lógicas sendo geralmente utilizados em expressões condicionais:
  - `&&` → AND (E lógico)
  - `||` → OR (OU lógico)
  - `!` → NOT (Operador de negação)



# Precedência de Operadores

- Precedência é a prioridade com que os operadores são executados pelo compilador. Caso os operadores tenham o mesmo nível de precedência eles são analisados da esquerda para a direita
- A precedência dos operadores pode ser mudada utilizando parênteses.

Prioridades dos operadores aritméticos	
Alta	Incremento(++), Decremento(--)
	Menos unitário(-)
	Multiplicação(*), Divisão(/), Resto da Divisão(%)
Baixa	Soma(+), Subtração(-)

# Operador cast

- Tem como função forçar para que uma expressão seja de um determinado tipo. Sua sintaxe é:

`(tipo de dado) expressão`

- Exemplo:

...

```
int dividendo=10, divisor=3;
```

```
float quociente=0.0;
```

```
quociente= (float)dividendo/divisor;
```

```
printf("%d/%d=%.2f\n",dividendo,divisor, quociente);
```

...

# Operadores bit a bit

- A linguagem C é considerada de baixo nível, pois permite a manipulação de bits:
  - `&` → AND
  - `|` → OR
  - `^` → XOR (OR exclusivo)
  - `~` → NOT (Complemento de um)
  - `<<` → Deslocamento para esquerda
  - `>>` → Deslocamento para direita