Aula 01 - Revisão C: Fundamentos

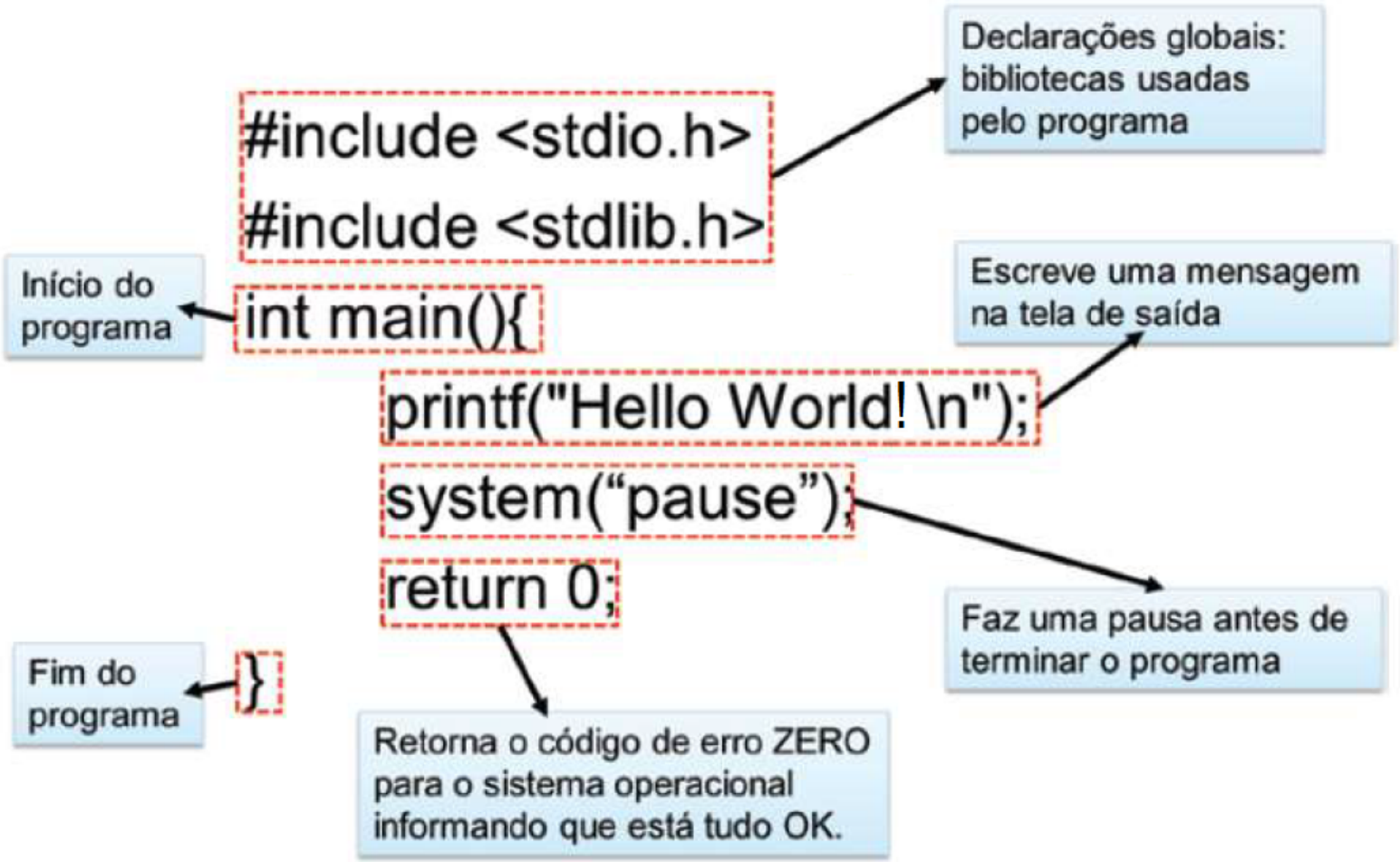
Prof. Me. Claudiney R. Tinoco

profclaudineytinoco@gmail.com

Faculdade de Computação (FACOM)
Bacharelado em Ciência da Computação (BCC)
Bacharelado em Sistemas de Informação (BSI)

Algoritmos e Estruturas de Dados 1 (AED1)
GBC024 - GSI006

Primeiro Programa em C



Comentários

Permitem adicionar uma descrição sobre o programa.
São ignorados pelo compilador.

*main.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      /* a função prinf() serve para escrever na
7      tela */
8
9      printf("Hello world!\n");
10
11     // Faz uma pausa no programa (em Windows)
12     system("pause");
13
14     // Retorna 0 (zero) para o sistema operacional
15     return 0; // FIM DO PROGRAMA
16 }
17
```

Comentário com uma ou mais linhas
Início: /*
Fim: */

Comentário em uma única linha
Início: //
Fim: sem símbolos, é o próprio final da linha

Variáveis

- Variável em programação:
  - Posição de memória que armazena uma informação;
  - Pode ser modificada pelo programa;
  - Deve ser **definida** antes de ser usada.
- Declaração de variável em C:
  `<tipo de dado> nome-da-variavel`

Regras de nomeação para variáveis:

- Nome:
  - Deve iniciar com letras ou underscore(_);
  - Caracteres devem ser letras, números ou underscores;
  - Palavras chaves não podem ser usadas como nomes;
  - Letras maiúsculas e minúsculas são consideradas diferentes (Case sensitive);
  - Não pode haver duas variáveis com o mesmo nome (no mesmo escopo).

Palavras reservadas da Linguagem

Lista de Palavras Reservadas (32)							
if	break	case	char	const	continue	default	do
else	while	for	float	goto	double	extern	auto
int	return	long	short	signed	sizeof	register	static
void	switch	union	enum	volatile	unsigned	typedef	struct

Tipos básicos em C

- **char**: um byte que armazena o código de um caractere do conjunto de caracteres local.

```
char sexo; // pode receber 'M' ou 'F'
char UnidadeTemperatura; //pode receber 'C' para Celsius
                         //ou 'F' para Fahrenheit
char opcoes; // pode ser '1', '2' ,'3' ou '4'
```

caracteres sempre ficam entre 'aspas simples'!

- **int**: um inteiro cujo tamanho depende do processador, tipicamente 16 ou 32 bits.

```
int NumeroAlunos;
int Idade;
int NumeroContaCorrente;
int N = 10; // o variável N recebe o valor 10
```

Tipos básicos em C

Números Reais

- **float**: um número real com precisão simples.

```
float Temperatura;    // pode receber, por exemplo, 23.30
float MediaNotas;     // pode receber, por exemplo, 7.98
float TempoTotal;     // pode receber 0.0000000032 (s) ou
                      // 3.2000e-009 (notação científica)
                      // que equivale à 3,2x10^(-9)
```

- **double**: um número real com precisão dupla.

```
double DistanciaGalaxias; // número muito grande
double MassaMolecular;    // em Kg, número muito pequeno
double BalancoEmpresa;    // valores financeiros
```

Tipos de Dados

Palavra chave	Tipo	Armazenamento	intervalo	Precisão
char	Caracter	1 byte	-128 to 127	
signed char	Caractere com sinal	1 byte	-128 to 127	
unsigned char	Caractere sem sinal	1 byte	0 to 255	
int	Inteiro	2 ou 4 bytes	-32,768 to 32,767 or -2,147,483,648 to 2,147,483,647	
signed int	Inteiro com sinal	2 ou 4 bytes	-32,768 to 32,767 or -2,147,483,648 to 2,147,483,647	
unsigned int	Inteiro sem sinal	2 ou 4 bytes	0 to 65,535 or 0 to 4,294,967,295	
short	Inteiro curto	2 bytes	-32,768 to 32,767	
signed short int	Inteiro curto com sinal	2 bytes	-32,768 to 32,767	
unsigned short	Inteiro curto sem sinal	2 bytes	0 to 65,535	
long	Inteiro longo	4 ou 8 bytes	-9223372036854775808 to 9223372036854775807	
signed long int	Inteiro longo com sinal	4 ou 8 bytes	-9223372036854775808 to 9223372036854775807	
unsigned long	Inteiro longo sem sinal	4 ou 8 bytes	0 to 18446744073709551615	
float	Ponto flutuante com precisão simples	4 bytes	1.2E-38 to 3.4E+38	6 casas dec.
double	Ponto flutuante com precisão dupla	8 bytes	2.3E-308 to 1.7E+308	15 casas dec.
long double	Ponto flutuante com precisão dupla longo	10 bytes	3.4E-4932 to 1.1E+4932	19 casas dec.

Constantes

- Como uma variável, uma constante também armazena um valor na memória do computador;
- Entretanto, esse valor não pode ser alterado: é constante;
- Para constantes é obrigatória a atribuição do valor.

Definindo constantes:

- Usando **#define**
  - Você deverá incluir a diretiva de pré-processador #define antes de início do código:
    ```
    #define PI 3.1415
    ```
- Usando **const**
  - Usando const, a declaração não precisa estar no início do código:
    ```
    const double pi = 3.1415;
    ```

Atribuição

- Operador de Atribuição: =
  `nome_da_variável = expressão, valor ou constante;`

Importante

O operador de atribuição "=" armazena o valor ou resultado de uma expressão contida a sua **direita** na variável especificada à sua **esquerda**.

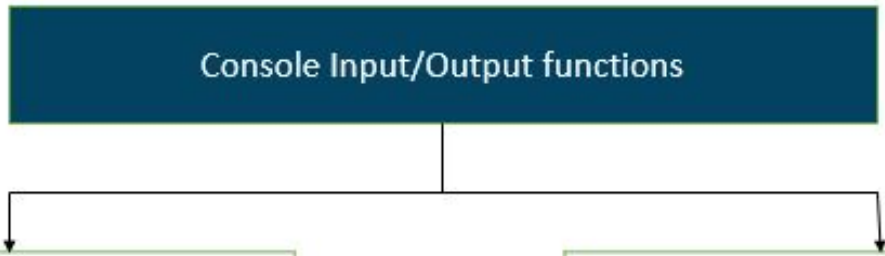
```
int main( ) {
    int x = 5; /* em pseudoliguagem
                  representamos assim: x <- 5 */
    int y;
    y = x + 3;
}
```

- A linguagem C suporta múltiplas atribuições
  `x = y = z = 0;`

Funções de E/S

As funções de entrada (=input) e de saída (=output) estão na biblioteca **stdio**. Para ter acesso a essa biblioteca, seu programa deve incluir a interface da biblioteca por meio de:

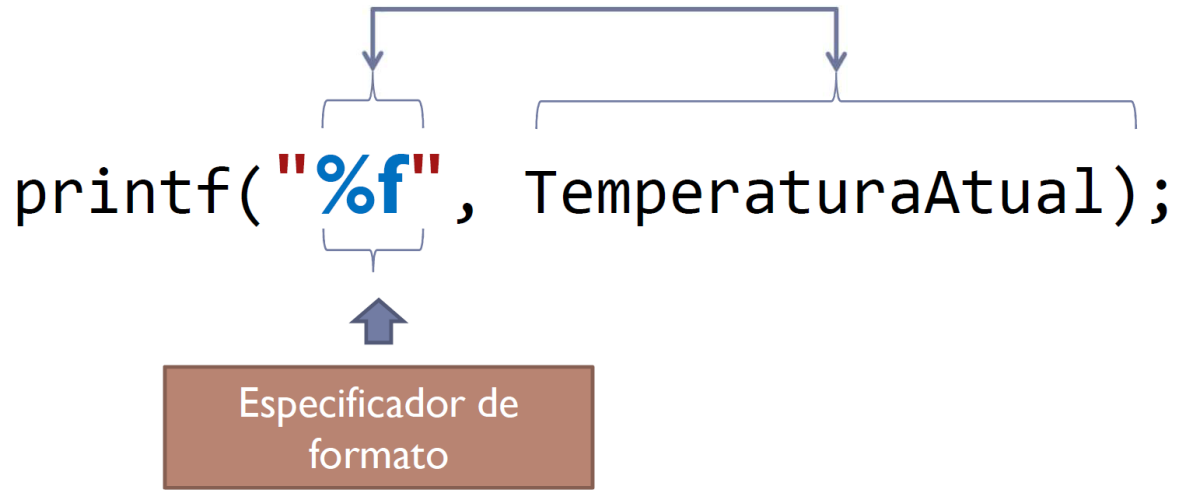
```
#include <stdio.h>
```



Formatted Input/Output function		
Type	**Input**	**Output**
char	scanf( )	printf( )
int	scanf( )	printf( )
float	scanf( )	printf( )
string	scanf( )	printf( )

Unformatted Input/Output function		
Type	**Input**	**Output**
char	getch( ) getche( ) getchar( )	putch( ) putchar( )
int	-	-
float	-	-
string	gets( )	puts( )

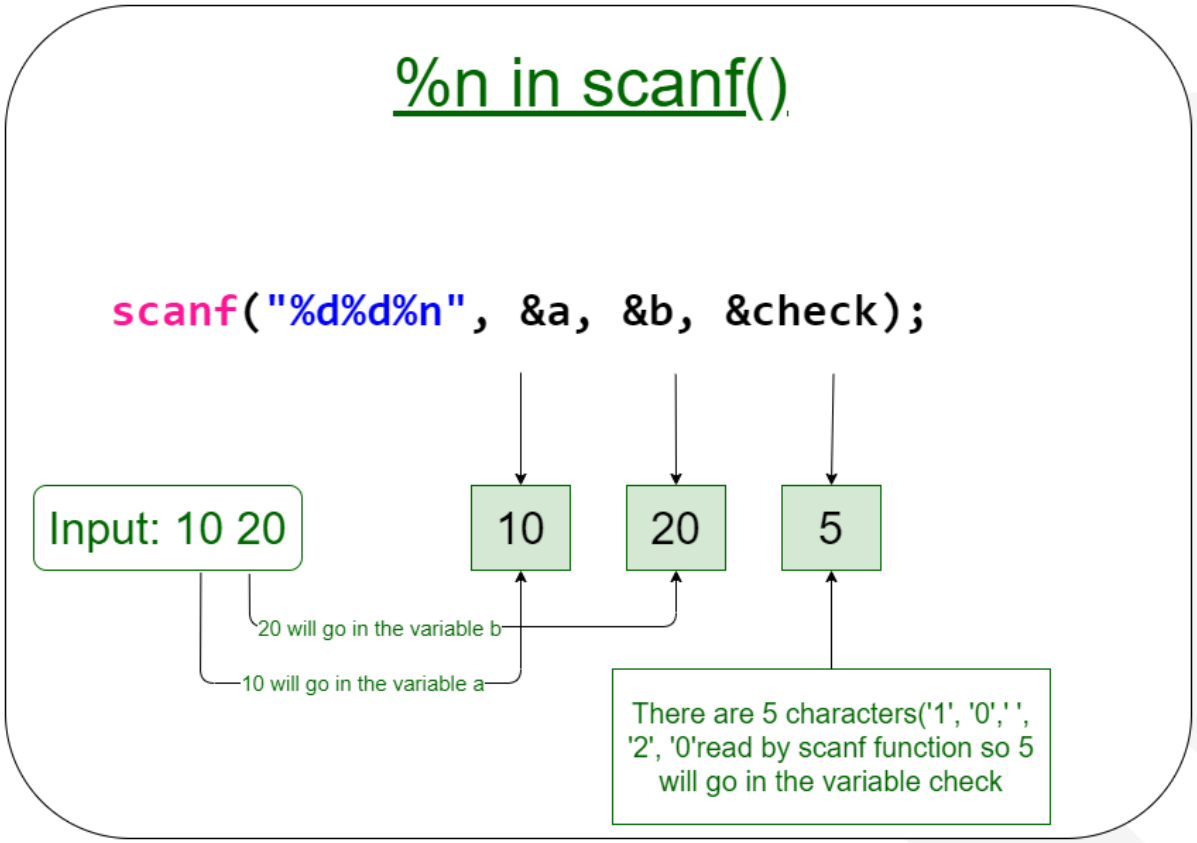
- **printf**



Alguns tipos de saída:

- %c – escrita de um caractere
- %d – escrita de números inteiros
- %f – escrita de número reais
- %s – escrita de vários caracteres

- **scanf**



Alguns tipos de entrada:

- %c – leitura de um caractere
- %d – leitura de números inteiros
- %f – leitura de float
- %s – leitura de vários caracteres

A linguagem C utiliza vários códigos chamados códigos de barra invertida.

Código	Significado
\b	Retrocesso ("back")
\a	Sinal sonoro ("beep")
\\	Barra invertida
\v	Tabulação vertical
\”	Aspas
\’	Apóstrofo
\0	Nulo (0 em decimal)
\n	Nova linha ("new line")
\t	Tabulação horizontal ("tab")
\f	Alimentação de formulário ("form feed")
\N	Constante octal (N é o valor da constante)
\xN	Constante hexadecimal (N é o valor da constante)

Limpando Buffer

- Antes de usar um scanf com "%c" faça
- **setbuf(stdin, NULL);**
- Esse comando limpa o buffer de entrada

```
char letra;
setbuf(stdin,NULL);
scanf("%c",&letra);
```

Modeladores (Casts)

Um modelador é aplicado à uma expressão.
Força o resultado da expressão a ser de um tipo específico.

(tipo) expressão

```
(float) x;
(int) x * 5.25;
```

Exemplo1:

```
int num;
float f;

num = 10;
f = num/7;
printf ("%f \n", f);

f = (float) num/7;
printf ("%f", f);
```

Resultado:

```
1.000000
1.428571
```

Exemplo2:

```c
int Inteiro;
float Real;

Real = 1/3;
printf("%f \n",Real); // resposta 0.00000

Real = 1/3.0;
printf("%f \n",Real); // resposta 0.33333

Inteiro = 3;
Real = 1/Inteiro;
printf("%f \n",Real); // resposta 0.00000

Real = 1/(float)Inteiro;
printf("%f \n",Real); // resposta 0.33333

Real = 2.9;
Inteiro = Real;
printf("%d \n",Inteiro); // resposta 2
```

Operadores

Um operador é um símbolo que indica a realização de uma operação sobre uma ou mais variáveis ou valores.

Operadores unários

Op	Uso	Exemplo
+	mais unário ou positivo	+x
-	menos unário (número oposto)	-x
!	NOT ou negação lógica	!x
&	Endereço	&x

Operadores binários

Operador	Descrição	Exemplo
+	Adição de dois números	`z = x + y;`
-	Subtração de dois números	`z = x - y;`
*	Multiplicação de dois números	`z = x * y;`
/	Quociente de dois números	`z = x / y;`
%	Resto da divisão	`z = x % y;`

Operadores relacionais

Realiza comparação entre variáveis. Esse tipo de operador retorna verdadeiro (1) ou falso (0).

Op	Descrição	Exemplo
>	Maior do que	`Idade > 6`
>=	Maior ou igual a	`Nota >= 60`
<	Menor do que	`Valor < Temperatura`
<=	Menor ou igual a	`Velocidade <= MAXIMO`
==	Igual a	`Opcao == 'a'`
!=	Diferente de	`Opcao != 's'`

Operadores lógicos

Operam com valores lógicos e retornam um valor lógico verdadeiro (1) ou falso (0).

Op	Função	Exemplo
&&	AND (E)	`(c >='0' && c<='9')`
||	OR (OU)	`(a=='F' || b!=32)`
!	NOT	`!continuar`

Importante

Símbolo de atribuição = é diferente, muito diferente, do operador relacional de igualdade ==

```c
int Nota;

Nota = 50; // Nota recebe 50

// Erro comum em C:
// Teste se a nota é 60
// Sempre entra na condição
if (Nota = 60) {
    printf("Você passou raspando!!");
}

// Versão Correta
if (Nota == 60) {
    printf("Você passou raspando!!");
}
```

Exercício

Faça um programa que leia a base e a altura de um triângulo. Em seguida, calcule e escreva (com 3 casas de precisão) a área do mesmo (Área = (Base*Altura)/2).

Controle de Fluxo

Comando **if**

- Em linguagem C, o comando if é utilizado quando for necessário escolher entre dois caminhos, ou quando se deseja executar um comando sujeito ao resultado de um teste.

- A forma geral de um comando if é:

```
if (expressão)
      instrução
```

- A expressão, na condição, será avaliada:
  - Se ela for zero (falsa), a instrução não será executada;
  - e a condição for diferente de zero (verdadeira) a instrução será executada.

Exemplo if

```c
int main()
{
    int num;

    printf("Digite um numero: ");
    scanf("%d", &num);

    if (num == 10) {
        printf("O numero eh igual a 10.\n");
    }
    return 0;
}
```

- Pode-se usar chaves { } para delimitar o bloco de instruções que pertence ao if

```
if (num > 10) {
  printf ("\n\n O numero eh maior que 10");
}
```

- As chaves **devem** ser usadas no caso de mais de uma instrução:

```
if (nota >= 60) {
    printf ("A nota é maior ou igual a 60 \n") ;
    printf ("O aluno está aprovado!") ;
}
```

- As chaves podem ser ignoradas se a instrução for única.

```
if (num > 10)
  printf ("\n\n O numero e maior que 10") ;
```

Comando **else**

- O comando else pode ser entendido como sendo um complemento do comando if.
  - Se o if diz o que fazer quando a condição é verdadeira, o else trata da condição falsa.

- O comando if-else tem a seguinte forma geral:

```
if (expressão)
      instrução1
else
      instrução2
```

Exemplo else

```c
int main()
{
    int num;

    printf("Digite um numero: ");
    scanf("%d", &num);

    if (num == 10){
        printf("O numero eh igual a 10.\n");
    } else {
        printf("O numero eh diferente de 10.\n");
    }
    return 0;
}
```

Comando **if-else**

Como no caso do comando if, as chaves podem ser ignoradas se a instrução contida no else for única.

```c
if (num==10){
   printf("O numero eh igual a 10.\n");
} else // else sem usar chaves
    printf("O numero eh diferente de 10.\n");
```

ou

```c
if (num==10){
    printf("O numero eh igual a 10.\n");
} else { // else com chaves
    printf("O numero eh diferente de 10.\n");
}
```

Aninhamento de **if-else**

- O if aninhado é simplesmente um if dentro da declaração de um outro if externo.
- O único cuidado que devemos ter é o de saber exatamente a qual if um determinado else está ligado.

```
//Classificação das notas
if( (nota>=8) && (nota<=10) )
    printf("Nota correspondente a A!!!\n");
else
    if( (nota>=6) && (nota<8) )
        printf("Nota correspondente a B!!!\n");
    else
        if( (nota>=4) && (nota<6) )
            printf("Nota correspondente a C!!!\n");
        else
            if( nota<4 )
                printf("Nota correspondente a D!!!\n");
            else
                printf("Nota invalida!!!\n");
```

Comando **switch**

- O comando switch é próprio para se testar uma variável em relação a diversos valores pré-estabelecidos.
  - Parecido com if-else-if, porém não aceitas expressões, apenas constantes.
  - O switch testa a variável e executa a declaração cujo case corresponda ao valor atual da variável.

Exemplo switch

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  int main(){
04    char ch;
05    printf("Digite um simbolo de pontuacao: ");
06    ch = getchar();
07    switch( ch ) {
08        case '.': printf("Ponto.\n" ); break;
09        case ',': printf("Virgula.\n" ); break;
10        case ':': printf("Dois pontos.\n" ); break;
11        case ';': printf("Ponto e virgula.\n"); break;
12        default : printf("Nao eh pontuacao.\n" );
13    }
14    system("pause");
15    return 0;
16  }
```

Características do **switch**

- O comando switch
  - Avalia o valor de expression com os valores associados às cláusulas case em sequencia;
  - Quando o valor associado a uma cláusula é igual ao valor de expression os respectivos comandos são executados até encontrar um break.
- A declaração default é opcional e será executada apenas se a expressão expression que está sendo testada não for igual a nenhuma das constantes presentes nos case.

Comando **if-else ternário - “? :”**

- A expressão condicional “? :” é uma simplificação do if-else, e é utilizada tipicamente para atribuições condicionais.

Exemplo if-else ternário

Dados dois números x e y, retorne o maior na variável z:

Usando if-else

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  int main(){
04    int x,y,z;
05    printf("Digite x:");
06    scanf("%d",&x);
07    printf("Digite y:");
08    scanf("%d",&y);
09    if(x > y)
10        z = x;
11    else
12        z = y;
13    printf("Maior = %d\n",z);
14    system("pause");
15    return 0;
16  }
```

Usando operador ternário

```
#include <stdio.h>
#include <stdlib.h>
int main(){
  int x,y,z;
  printf("Digite x:");
  scanf("%d",&x);
  printf("Digite y:");
  scanf("%d",&y);
  z = x > y ? x : y;
  printf("Maior = %d\n",z);
  system("pause");
  return 0;
}
```

Exercício

Suponha que uma empresa decidiu dar um aumento escalonado a seus funcionários de acordo com a seguinte regra:

- 13% para os salários inferiores ou iguais a R$ 200,00;
- 11% para os salários situados entre R$ 200,00 e R$ 400,00 (inclusive);
- 9% para os salários entre R$ 400,00 e R$ 800,00 (inclusive); e
- 7% para os demais salários.

Faça um programa que receba o salário atual de um funcionário e forneça o valor do seu novo salário. *(utilizar comandos if aninhados)*

Laços

Definição:

Uma estrutura de repetição (laço) permite que uma sequência de comandos seja executada repetidamente, enquanto determinadas condições sejam satisfeitas. Essas condições são representadas por expressões lógica [como, por exemplo: $(A > B)$; (C==3); (letra == 'a')]

Tipos de estruturas de repetição:

- Repetição com Teste no Início (**while**);
- Repetição com Teste no Final (**do - while**);
- Repetição Contada (**for**)

Comando **while**

- Repede a sequencia de comandos enquanto a condição for verdadeira.
- Esse comando possui a seguinte forma geral:

```
while (expressão)
      instrução1
```

Exemplo while

Faça um programa que mostra na tela os número de 1 a 100:

```c
int main()
{
    // programa que mostra na tela números de 1 ate 100
    int numero;

    numero = 1;
    while (numero <= 100){
        printf(" %d ", numero);
        numero = numero + 1;
    }

    return 0;
}
```

Comando **do-while**

- Comando while: é utilizado para repetir um conjunto de comandos zero ou mais vezes.
- Comando do-while: é utilizado sempre que o bloco de comandos **deve ser executado ao menos uma vez.**

Exemplo do-while

Faça um menu simples, com três opções:

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

int main()
{
    int i;
    do{
        printf("Escolha uma opcao:\n");
        printf("(1) Opcao 1 \n");
        printf("(2) Opcao 2 \n");
        printf("(3) Opcao 3 \n");
        scanf("%d", &i);
    } while ((i<1)||(i>3));
    printf("Opcao escolhida: %d \n",i);
    return 0;
}
```

Comando **for**

- O loop ou laço for é usado para repetir um comando, ou bloco de comandos, diversas vezes.
  - Maior controle sobre o loop.
- Esse comando possui a seguinte forma geral:

```
for (inicialização; condição; incremento)
    instrução1
```

  - inicialização: iniciar variáveis (contador);
  - condição: avalia a condição. Se verdadeiro, executa comandos do bloco, senão encerra laço;
  - incremento: ao término do bloco de comandos, incrementa o valor do contador;
  - *repete o processo até que a condição seja falsa.*

Exemplo for

Escreva, usando for, um algoritmo para calcular a soma dos números inteiros de 1 a 10.

```
int n;
int soma = 0;
for (n = 1; n <= 10; n++){
      soma = soma + n;
}
printf("%d",soma);
```

1

Comando for

- Podemos omitir qualquer um de seus elementos
  - inicialização, condição ou incremento.
- Cuidado: for sem condição
  - omitir a condição cria um laço infinito;
  - condição será sempre verdadeira.
- Cuidado: for sem incremento
  - omitir o incremento pode criar um laço infinito;
  - incremento pode ser feito nos comandos.

Comandos Especiais **break** e **continue**

- Comando break:
  - Quebra a execução de um comando (como no caso do switch);
  - Interrompe a execução de qualquer estrutura de repetição;
  - Para a execução do programa;
  - A execução do programa continua na primeira linha seguinte ao loop ou ao bloco que está sendo interrompido.
- Comando continue:
  - Funciona apenas dentro de um loop;
  - Pula para a próxima iteração do loop, sem o abandono do mesmo.

Exercício

Utilizando o comando for, fazer um programa para encontrar todos os números pares entre 1 e 100.

Referências

✓ Básica

- **DAMAS, Luís. "Linguagem C". Grupo Gen-LTC, 2016.**
- **MIZRAHI, Victorine V. "Treinamento em linguagem C", 2a. ed., São Paulo, Pearson, 2008.**

✓ Extra

- BACKES, André. *"Programação Descomplicada Linguagem C"*. Projeto de extensão que disponibiliza vídeo-aulas de C e Estruturas de Dados. Disponível em: `https://www.youtube.com/user/progdescomplicada`. Acessado em: 25/04/2022.

✓ Baseado nos materiais dos seguintes professores:

- Prof. André Backes (UFU)
- Prof. Bruno Travençolo (UFU)
- Prof. Luiz Gustavo de Almeida Martins (UFU)

Dúvidas?

Prof. Me. Claudiney R. Tinoco
profclaudineytinoco@gmail.com

Faculdade de Computação (FACOM)
Universidade Federal de Uberlândia (UFU)