

Tecnologia em Análise e Desenvolvimento de Sistemas - TADS

Estrutura de Dados

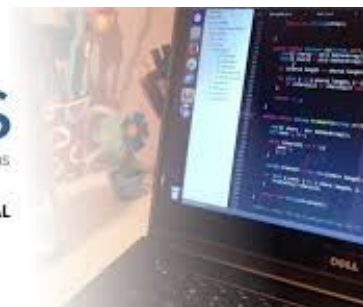
TADS / IFRS - 2025-1


Prof. Luciano Vargas Gonçalves

E-mail: luciano.goncalves@riogrande.ifrs.edu.br



INSTITUTO FEDERAL
Rio Grande do Sul
Campus Rio Grande



- 
- Ambiente de Programação
 - Linguagem C
 - Conceitos iniciais

Linguagem C e Compilador C

- Tutoriais para Leitura e Consulta sobre a linguagem C
 - <http://www.inf.ufrgs.br/~binsely/tutorialc.pdf>
 - https://www.cenapad.unicamp.br/servicos/treinamentos/apostilas/apostila_C.pdf

Linguagem C e Compilador C

- Instalação e Configuração Linux e Windows
 - Linux
 - [Link 1 – Ubuntu 18.04](#)
 - [Link 2 – Vídeo](#)
 - Windows
 - [Link 1 - Tutorial para Windows 10](#)
 - [Link 2 – Vídeo para instalar o MinGW](#)

Editar e compilar um código C

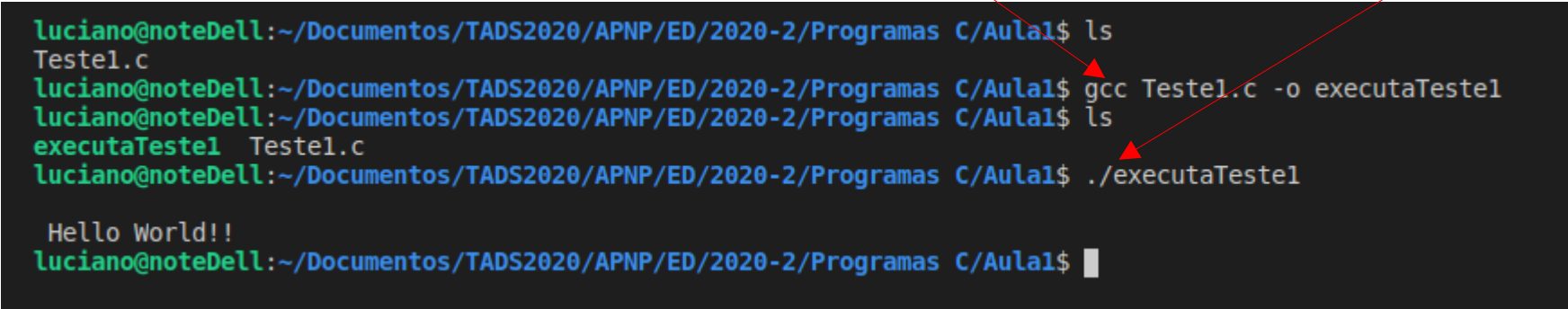
- Código mínimo – Hello World!!

- Editor VsCode
- Arquivo Teste.c
 - #Include < >
 - Função main()

```
Programas C > Aula1 > C Teste1.c > ...
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      printf("\n Hello World!! \n");
7  }
8
```

Editar e compilar um código C

- Código mínimo – Hello World!!
 - Compilar no Terminal.
 - Acessar a pasta “Aula1” e o arquivo fonte “Teste.c”.
 - Compilar o arquivo com o compilador **gcc** do linux, executar o arquivo com “./”



```
luciano@noteDell:~/Documentos/TADS2020/APNP/ED/2020-2/Programas C/Aula1$ ls
Teste1.c
luciano@noteDell:~/Documentos/TADS2020/APNP/ED/2020-2/Programas C/Aula1$ gcc Teste1.c -o executaTeste1
luciano@noteDell:~/Documentos/TADS2020/APNP/ED/2020-2/Programas C/Aula1$ ls
executaTeste1  Teste1.c
luciano@noteDell:~/Documentos/TADS2020/APNP/ED/2020-2/Programas C/Aula1$ ./executaTeste1

Hello World!!
luciano@noteDell:~/Documentos/TADS2020/APNP/ED/2020-2/Programas C/Aula1$
```

Olá Mundo em C

Linguagem C é Compilada (Linguagem máquina)

- Para compilar: (verifica a sintaxe)

```
>> gcc fonte.c <ENTER>
```

- Para compilar e gerar executável:

```
>> gcc fonte.c -o nomeexec <ENTER>
```

- Executando:

```
>> ./nomeexec <ENTER>
```

OBS: Se o nome do executável não for informado o default é *a.out*

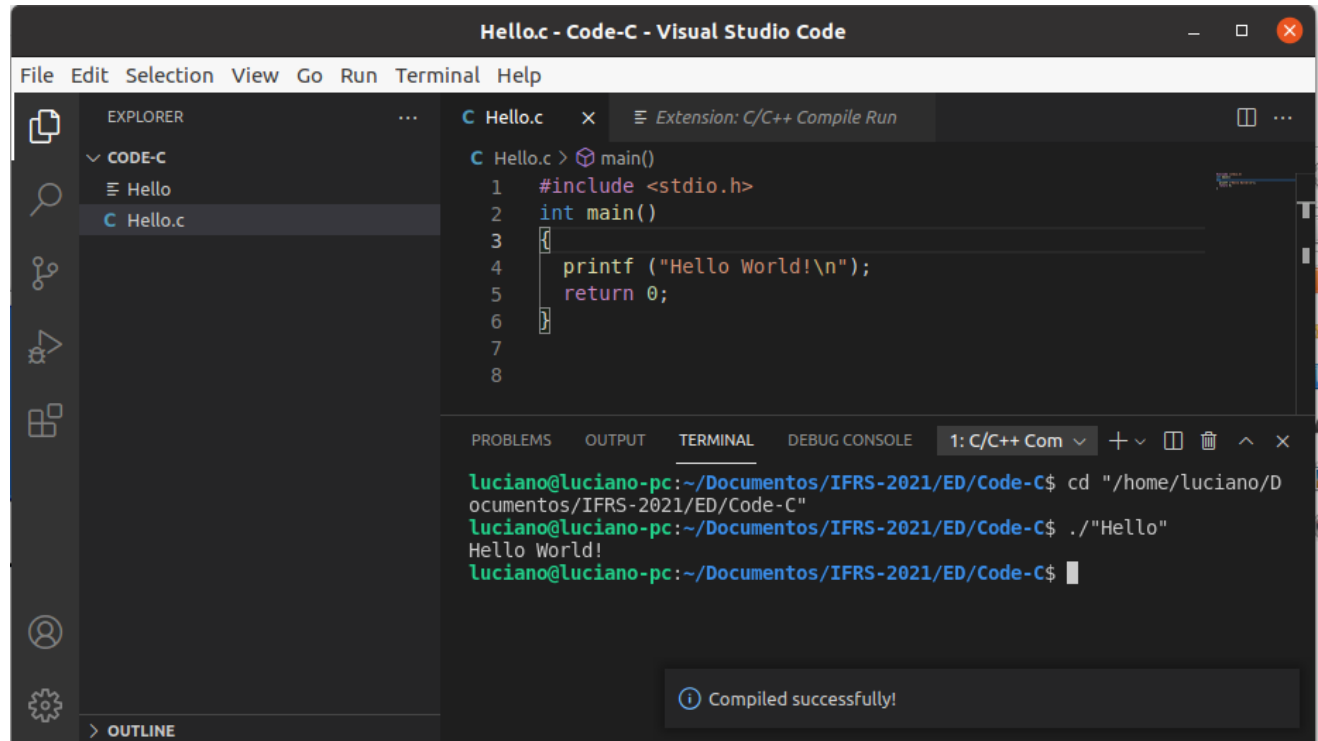
Instalação do VsCode:

- Download - VsCode
 - <https://code.visualstudio.com/>
- Tutorial de instalação
 - Windows
 - <https://code.visualstudio.com/docs/languages/cpp>
 - Linux
 - <https://sempreupdate.com.br/como-instalar-o-visual-studio-code-no-ubuntu-20-04/>



Instalação do VsCode:

- Precisa criar uma pasta e um arquivo Hello.c
- Tecla F6
 - Compilar



The screenshot shows the Visual Studio Code interface with a C program named 'Hello.c' open. The Explorer sidebar on the left shows the project structure with a folder 'CODE-C' containing 'Hello' and 'Hello.c'. The main editor displays the code for 'Hello.c', which includes a standard C 'Hello World' program. The Terminal panel at the bottom shows the execution of the program, with the output 'Hello World!'. A status bar at the bottom right indicates 'Compiled successfully!'.

```
>Hello.c - Code-C - Visual Studio Code
File Edit Selection View Go Run Terminal Help

EXPLORER
CODE-C
  Hello
  C Hello.c

C Hello.c
1 #include <stdio.h>
2 int main()
3 {
4     printf ("Hello World!\n");
5     return 0;
6 }
7
8

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE 1: C/C++ Com
Luciano@luciano-pc:~/Documentos/IFRS-2021/ED/Code-C$ cd "/home/luciano/D
ocumentos/IFRS-2021/ED/Code-C"
Luciano@luciano-pc:~/Documentos/IFRS-2021/ED/Code-C$ ./"Hello"
Hello World!
Luciano@luciano-pc:~/Documentos/IFRS-2021/ED/Code-C$

Compiled successfully!
```

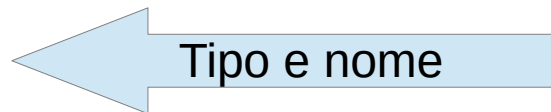
Introdução a Programação em C

- Faça uma revisão de algoritmos e implementações usando a linguagem C;
 - Avalie a Sintaxe da linguagem (estruturas);
 - Avalie a Semântica da linguagem (operações);
 - Livro online:
 - <http://www.inf.ufpr.br/lesoliveira/download/c-completo-total.pdf>
 - Desenvolva algoritmos com C
 - Exemplos e exercícios Online
 - Revisar até a aula de Strings
 - <https://programacaodescomplicada.wordpress.com/indice/linguagem-c/>
 - <https://www.youtube.com/c/excriptvideo/search?query=programa%C3%A7%C3%A3o%20C>

Revisão da Linguagem C

- **Reserva de Memória - Declaração;**
 - Declaração de uma variável é a alocação de espaço de memória, com um ***certo tipo de dado*** associado, e um nome para referenciar seu conteúdo.
 - Em C a **Tipagem forte**, necessita declaração de tipo.

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



Linguagem C

- **Nome de Variáveis**

- Quantos caracteres quiser (32);
- Comece com letras ou sublinhado:
 - Seguidos de letras, números ou sublinhados
- Linguagem ***C é sensível*** ao caixa:
 peso # Peso # pEso
- Não podemos definir um identificador com o mesmo nome que uma palavra-chave
 - ***Auto, static, extern, int, long, if, while, do***

Linguagem C

Declaração de Variáveis – Tipos primitivos, Tamanhos e Limites

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

Long ou Long int (4 bytes)
Unsigned Char (0 a 255)
Unsigned int (0 a 65.535)

Linguagem C

Declaração de Variáveis – Tipos primitivos

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

```
int main(){
```

declaração

```
{
    int soma = 10;
    float money = 2.21;
    char letra = 'A';
    double pi = 3.1415E6;
```

saída

```
{
    printf("\n valor da Soma = %d\n",soma);
    printf("\n valor de Money = %f\n",money);
    printf("\n valor da Letra = %c e o codigo ASC = %d\n",letra,letra);
    printf("\n valor da Pi = %e\n",pi);
```

```
}
```

Formatação da
Saída no
Terminal -Tela



Entrada de Dados

- Leitura de dados tipados via teclado

– *Scanf ("stringa de controle", lista de argumentos):*

Exemplo:

scanf("%d",&idade);

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

```
4
5
6 printf("\nInforme sua idade:");
7 //entrada de dados com scanf
8 scanf("%d",&idade);
9 printf("Sua idade é: %d \n",idade);
10 }
```

Formatação de Entrada do Terminal

Entrada de Dados

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

```
4  int main(){
5      int idade;
6      printf("\nInforme sua idade:");
7      //entrada de dados com scanf
8      scanf("%d",&idade);
9      printf("Sua idade é: %d \n",idade);
10 }
```

Saída no Terminal

• `luciano@luciano-pc:~/Documentos/IFRS2022/Disciplinas/Primeiro/Ed1/Exemplos/Aula1$./"Exemplo1"`

```
Informe sua idade:45
Sua idade é: 45
```


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**

USADOS NOS COMANDOS:

- **SCANF**
- **PRINTF**

Entrada e Saída de Dados

Exemplos de usos dos caracteres especiais de formatação

```
int main(){  
    char ch;  
    printf("Digite um character:");  
    scanf("%c",&ch);  
    printf(" %c = %d em decimal Asc ",ch,ch);  
    printf("\n %c = %o em octal, %x em hexadecimal\n ",ch,ch,ch);  
}
```

Digite um character:A

A = 65 em decimal Asc

A = 101 em octal, 41 em hexadecimal

Saída no
Terminal

Endereço de memória

- **Nome e o Tipo** de uma variável é necessário para o programador solicitar ao computador(compilador) a reserva de memória para armazenar uma ou mais informações;
- O nome da variável só é importante para o programador localizar a informação no seu programa;
- O computador usa para identificar uma variável ou espaço da memória, o ***seu endereço de memória***;
- Toda variável ocupa **uma porção** de memória conforme seu tipo de dados. Para cada porção de memória o primeiro byte ocupado por ela é o seu endereço na memória (RAM);

Endereço de Memória

- Exemplo: variável IDADE (int) ocupa 4bytes

- 68e99724 é o seu endereço

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

```
int main(){
    int idade;
    printf("\n Informe sua idade:");
    scanf("%d",&idade);

    printf("\n Sua idade é: %d\n",&idade);
    printf("\n IDADE esta armazenada memoria = %x\n",&idade);
}
```

Informe sua idade:45

Saída no terminal

Sua idade é: 45

IDADE esta armazenada memoria = 68e99724

Endereço	Informação
68e99722	
68e99723	
68e99724	45
68e99725	
68e99726	
68e99727	
68e99728	

Endereço de Memória

- Exemplo: idade, idade2 ocupam diferentes posições de memória, espaçadas 4Bytes

```
int main(){
    int idade,idade2;
    printf("\nInforme sua idade:");
    scanf("%d",&idade);
    printf("\nInforme idade seu irmão:");
    scanf("%d",&idade2);
    printf("Sua idade é: %d seu irmão é: %d \n",idade,idade2);

    printf("Variável idade está armazenada na posição de memória %x \n",&idade);
    printf("Variável idade2 está armazenada na posição de memória %x \n",&idade2);
}
```

Informe sua idade:45

Informe idade seu irmão:38

Sua idade é: 45 seu irmão é: 38

Variável idade está armazenada na posição de memória d2559960

Variável idade2 está armazenada na posição de memória d2559964

Endereço	Informação
d2559960	45
d2559961	
d2559962	
d2559963	
d2559964	38
d2559965	
d2559966	
d2559967	

Atribuição de Endereço de memória "&"

- Quando usamos o operador "&" para acesso ou atribuição do valor do **endereço de memória**;
- Exemplo:
 - Usado no Scanf, para atribuir um valor a uma posição de memória

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



Atribuição de Endereço de memória

"&"

- Quando usamos o operador "&" para acesso e atribuição à endereços de memória;
- Exemplo:
 - Usado para consultar o endereço de uma variável:

```
printf("Variável idade está armazenada na posição de memória %x \n",&idade);
```



Consulta o Endereço da
variável

Saída de Dados

- Comando printf formatação para REAIS

```
✓ int main(){  
    int total = 350;  
    //saida de dados "printf()"  
    printf("o n° de aluno são %d \n",total);  
    printf("o n° de aluno são %2d \n",total);  
    printf("o n° de aluno são %4d \n",total);  
    printf("o n° de aluno são %10d \n",total);  
}
```

```
o n° de aluno são 350  
o n° de aluno são 350  
o n° de aluno são  350  
o n° de aluno são          350
```

Ocupa 10 posições

Saída de Dados

- Comando printf formatação para REAIS

```
int main(){  
    double total = 35.123456789;  
    //saida de dados "printf()"   
    printf("o n° de aluno são %f \n",total);  
    printf("o n° de aluno são %2.2f \n",total);  
    printf("o n° de aluno são %2.4f \n",total);  
    printf("o n° de aluno são %4.10f \n",total);  
}
```

```
o n° de aluno são 35.123457  
o n° de aluno são 35.12  
o n° de aluno são 35.1235  
o n° de aluno são 35.1234567890
```

Ocupa 10 posições após a vírgula

Saída de Dados

- Comando printf formatação para REAIS

```
✓ int main(){  
    float total = 35.123456789;  
    //saida de dados "printf()"   
    printf("o n° de aluno são %f \n",total);  
    printf("o n° de aluno são %2.2f \n",total);  
    printf("o n° de aluno são %2.4f \n",total);  
    printf("o n° de aluno são %4.10f \n",total);  
}
```

```
o n° de aluno são 35.123455  
o n° de aluno são 35.12  
o n° de aluno são 35.1235  
o n° de aluno são 35.1234550476
```

Ocupa 10 posições após a vírgula

Saída de Dados

- Float vs Double precisão de representação

```
int main(){
    float a = 35.111111111111;
    float b = 35.111111111111;
    float c=a+b;
    //saida de dados "printf()"
    printf("A %4.20f \n",a);
    printf("B %4.20f \n",b);
    printf("C %4.20f \n",c);
```

```
A 35.11111068725585937500
B 35.11111068725585937500
C 70.22222137451171875000
```

Erro computacional -
Valor real para binário!!

```
int main(){
    double a = 35.111111111111;
    double b = 35.111111111111;
    double c=a+b;
    //saida de dados "printf()"
    printf("A %4.20f \n",a);
    printf("B %4.20f \n",b);
    printf("C %4.20f \n",c);
```

```
A 35.11111111111100058224
B 35.11111111111100058224
C 70.22222222222200116448
```

Maior precisão nos cálculos

Exemplo – Array de Char (String)

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

```
C Hello.c (deleted) C Exemplo1.c •
Aula1 > C Exemplo1.c > main()
1  #include <stdio.h>
2  int main()
3  {
4      int idade;
5      char nome[30];
6      printf("Digite o seu nome:");
7      scanf("%s",nome);
8      printf("Informe sua Idade:");
9      scanf("%d",&idade);
10     printf("Pessoa %s possui %d anos de idade. \n",nome,idade);
11     return 0;
12 }
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
luciano@luciano-pc:~/Documentos/IFRS-2021/ED/Code-C/Aula1$ ./"Exemplo1"
Digite o seu nome:Joao
Informe sua Idade:34
Pessoa Joao possui 34 anos de idade.
luciano@luciano-pc:~/Documentos/IFRS-2021/ED/Code-C/Aula1$
```

String e Vetor não usam "&"
Tipos compostos:
Ocupam várias posições
de memória

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:

- 0 (ZERO) representa FALSO
- 1 (UM ou maior que 1) representa VERDADE

```
Aula1 > C Exemplo2.c > main()
3  {
4      int teste1, teste2;
5
6      teste1 = 12 > 10;
7      teste2 = 12 < 10;
8
9      printf("Testes 1 resultado %d\n", teste1);
10     printf("Testes 2 resultado %d\n", teste2);
11
12     if(teste1)
13         printf("Verdade = %d \n", teste1);
14     else
15         printf("Falso = %d \n", teste1);
16
17     if(teste2)
18         printf("Verdade = %d \n", teste2);
19     else
20         printf("Falso = %d \n", teste2);
21     return 0;
22 }
```

PROBLEMS OUTPUT **TERMINAL** DEBUG CONSOLE 2: C/C++ Com ▾ + ▾ □ ✕ ^

```
luciano@luciano-pc:~/Documentos/IFRS-2021/ED/Code-C/Aula1$ ./"Exemplo2"
Testes 1 resultado 1
Testes 2 resultado 0
Verdade = 1
Falso = 0
```

Comparação

Pré Incremento ++X;

Pós incremento X++;

```
Aula1 > C Exemplo3.c > main()
4  int c = 0, d =0;
5
6  printf("Valor de C= %d\n",++c);
7  //incremente apos atribui
8  printf("Valor de C= %d\n",c);
9
10 printf("Valor de D= %d\n",d++);
11 //atribui e apos incrementa
12 printf("Valor de D= %d\n",d);
13 return 0;
14
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
luciano@luciano-pc:~/Documentos/IFRS-2021/E
Valor de C= 1
Valor de C= 1
Valor de D= 0
Valor de D= 1
luciano@luciano-pc:~/Documentos/IFRS-2021/E
```


Comparação

```
int main(){  
    int teste = (10>4 && !(10<9) || 3<=4);  
    printf("\n Resultado teste %d \n",teste);  
}
```

Resultado teste 1

```
int main(){  
    int teste = (10>4 && 10<9 || 3>=4);  
    printf("\n Resultado teste %d \n",teste);  
}
```

Resultado teste 0

Operador SIZEOF

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

Aula1 > C Exemplo4.c > main()

```
1  #include <stdio.h>
2  int main()
3  {
4      int a = 0;
5      float b = 10.04f;
6      char c = 'D';
7      double d = 12.565;
8
9      printf("Inteiro A= %d \t\t0cupa  %d Bytes\n",a, sizeof(a));
10     printf("Float B= %f \t\t0cupa  %d Bytes \n",b, sizeof(b));
11     printf("Char C= %c \t\t0cupa  %d Bytes \n",c, sizeof(c));
12     printf("Double D= %f \t\t0cupa  %d Bytes \n",d, sizeof(d));
13
14     return 0;
15 }
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
luciano@luciano-pc:~/Documentos/IFRS-2021/ED/Code-C/Aula1$ ./"Exemplo4"
Inteiro A= 0                Ocupa  4 Bytes
Float B= 10.040000          Ocupa  4 Bytes
Char C= D                  Ocupa  1 Bytes
Double D= 12.565000         Ocupa  8 Bytes
luciano@luciano-pc:~/Documentos/IFRS-2021/ED/Code-C/Aula1$
```

Operadores de Seleção - IF

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

Operadores de Seleção - IF

- IF/ELSE cascata

```
int main(){  
    int a = 2;  
    char op;  
    if( a==0)  
        printf("\nA igual a ZERO\n");  
    else if( a==1)  
        printf("\nA igual a UM\n");  
    else  
        printf("\nA diferente de Zero e UM\n");  
}
```

Operadores de Seleção - Switch

- Comando Switch / Case

```
int main(){
    int a = 2;
    switch (a){
        case 0:
            printf("\nA igual a ZERO\n");
            break;
        case 1:
            printf("\nA igual a UM\n");
            break;
        default:
            printf("\nA diferente de Zero e UM\n");
    }
}
```

Operadores de Seleção – Ternário ?

- Operador Ternário ?
 - Forma compacta de expressar uma instrução if – eles
 - (condição) ? expressão1 : expressão2;

Exemplo:

```
int main(){
    int a = 2;

    (a==0)?printf("\nA igual a ZERO\n"):(a==1)?printf("\nA igual a UM\n"):printf("\nA Diferente de Zero e UM\n");

    /*Equivalente e
    (a==0)?
        printf("\nA igual a ZERO\n")
        :
        (a==1)?
            printf("\nA igual a UM\n")
            :
            printf("\nA Diferente de Zero e UM\n");
    */
}
```

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

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

Exemplo com duas variáveis de condições

Aula1 > C Exemplo7.c > main()

```
1  #include <stdio.h>  
2  int main()  
3  {  
4      int x,y;  
5      for(x=0, y=0; x+y<100; ++x, y=x+y )  
6          printf("X=%d Y=%d = X+Y=%d\n",x,y,x+y);  
7  
8      return 0;  
9  }
```

PROBLEMS

OUTPUT

TERMINAL

DEBUG CONSOLE

```
luciano@luciano-pc:~/Documentos/IFRS-2021/ED/Code-C/Aula1$ ./"E  
X=0 Y=0 = X+Y=0  
X=1 Y=1 = X+Y=2  
X=2 Y=3 = X+Y=5  
X=3 Y=6 = X+Y=9  
X=4 Y=10 = X+Y=14  
X=5 Y=15 = X+Y=20  
X=6 Y=21 = X+Y=27  
X=7 Y=28 = X+Y=35  
X=8 Y=36 = X+Y=44  
X=9 Y=45 = X+Y=54  
X=10 Y=55 = X+Y=65  
X=11 Y=66 = X+Y=77  
X=12 Y=78 = X+Y=90  
luciano@luciano-pc:~/Documentos/IFRS-2021/ED/Code-C/Aula1$
```


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);  
}
```

Ponteiros e Structs

- **Para próxima Aula:**
 - **Leiam e testem os exemplo da material até a seção 7**
 - <http://www.inf.ufrgs.br/~binsely/tutorialc.pdf>