

# Aula 3

---

## Linguagem Técnica de Programação

Prof. Salatier Luz Marinho  
[salatier.marinho@docente.unip.br](mailto:salatier.marinho@docente.unip.br)

# Palavra-Chave

Há certas palavras reservadas para realizar tarefas específicas nos programas em C. Essas palavras são conhecidas como palavra reservada ou palavras-chave. Essas palavras são predefinidas e sempre devem ser escritas em letras minúsculas. Essas palavras-chave não podem ser usadas como sendo o nome de uma variável com significado fixo. De acordo SOFFNER (2013), a Linguagem C possui um total de 32 palavras reservadas, conforme definido pelo padrão ANSI. Segue a representação abaixo:

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

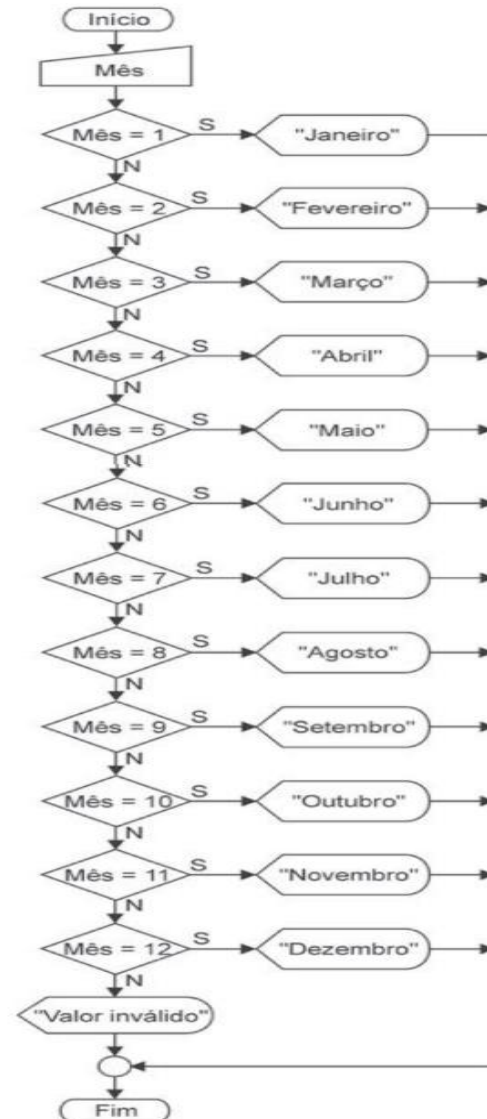
# Especificadores de Formato

Os especificadores de formato, que, como o nome diz, formatam a entrada e a saída-padrão. São eles:

- %d = inteiro decimal
- %f = real (ponto flutuante)
- %o = inteiro octal
- %x = hexadecimal
- %c = caractere em formato ASCII
- %s = string de caracteres
- %p = valor ponteiro

# Relembrar Não Custa Nada.....

Diagrama de Blocos



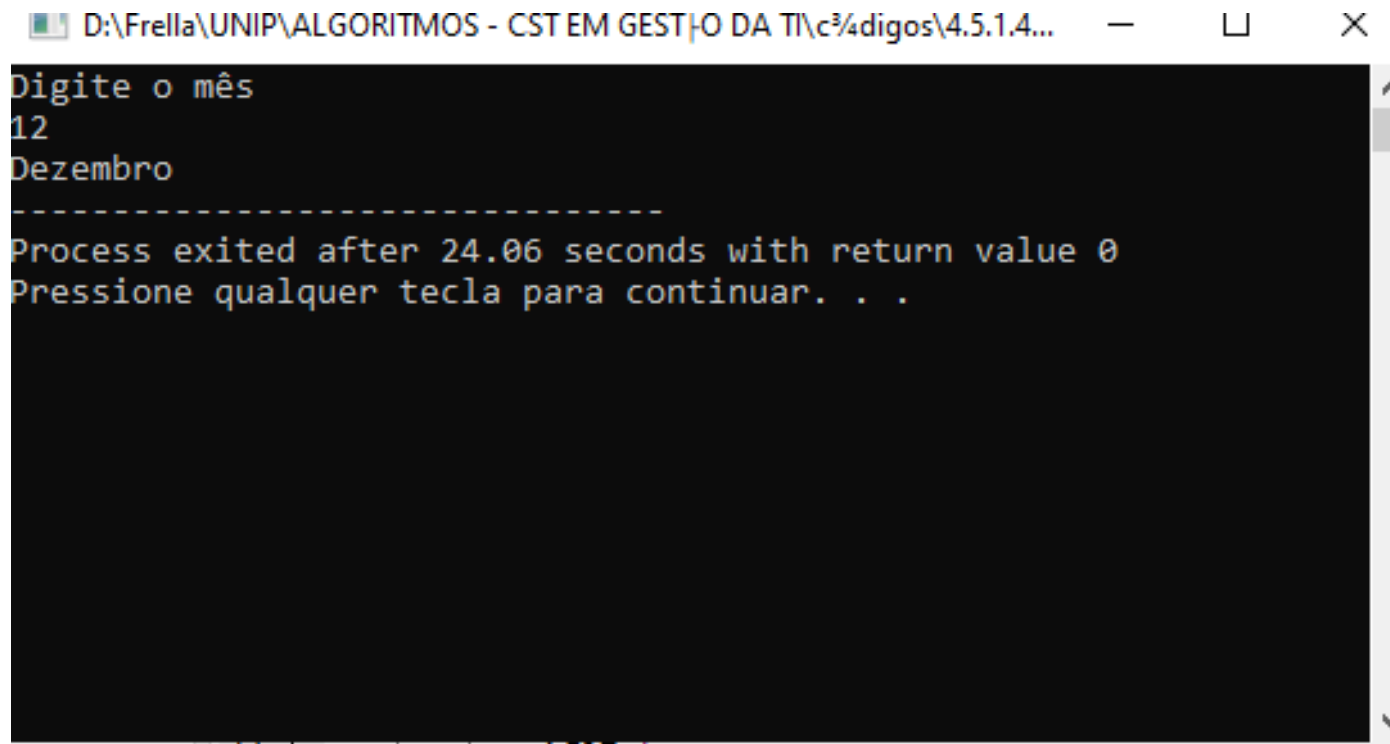
Português Estruturado

```
programa MÊS_POR_EXTENSO
var
  MÊS : inteiro
início
  leia MÊS
  caso MÊS
    seja 1 faça
      escreva "Janeiro"
    seja 2 faça
      escreva "Fevereiro"
    seja 3 faça
      escreva "Março"
    seja 4 faça
      escreva "Abril"
    seja 5 faça
      escreva "Maio"
    seja 6 faça
      escreva "Junho"
    seja 7 faça
      escreva "Julho"
    seja 8 faça
      escreva "Agosto"
    seja 9 faça
      escreva "Setembro"
    seja 10 faça
      escreva "Outubro"
    seja 11 faça
      escreva "Novembro"
    seja 12 faça
      escreva "Dezembro"
  senão
    escreva "Valor inválido"
  fim_caso
fim
```

# Relembrar Não Custa Nada.....

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <locale.h> //necessário para usar setlocale
4
5  /* ----- Estrutura else if ----- */
6  int main (void) {
7
8      //Utilizando caracteres e acentuação da língua portuguesa
9      setlocale(LC_ALL, "Portuguese");
10
11     int mes;
12
13     printf ("Digite o mês\n");
14     scanf ("%d",&mes);
15
16     switch (mes) {
17         case 1:
18             printf("Janeiro");
19             break;
20         case 2:
21             printf("Fevereiro");
22             break;
23         case 3:
24             printf("Março");
25             break;
26         case 4:
27             printf("Abril");
28             break;
29         case 5:
30             printf("Maio");
31             break;
32         case 6:
33             printf("Junho");
34             break;
35         case 7:
36             printf("Julho");
37             break;
38         case 8:
39             printf("Agosto");
40             break;
41         case 9:
42             printf("Setembro");
43             break;
44         case 10:
45             printf("Outubro");
46             break;
47         case 11:
48             printf("Novembro");
49             break;
50         case 12:
51             printf("Dezembro");
52             break;
53         default:
54             printf("Valor inválido");
55             break;
56     }
57     return 0;
58 }
```

# Relembrar Não Custa Nada.....



A screenshot of a Windows command prompt window. The title bar shows the file path: D:\Frella\UNIP\ALGORITMOS - CST EM GESTÃO DA TI\c\4digos\4.5.1.4... The window contains the following text: "Digite o mês", "12", "Dezembro", a dashed line separator, "Process exited after 24.06 seconds with return value 0", and "Pressione qualquer tecla para continuar. . .". The text is displayed in a monospaced font on a black background.

```
D:\Frella\UNIP\ALGORITMOS - CST EM GESTÃO DA TI\c\4digos\4.5.1.4...  
Digite o mês  
12  
Dezembro  
-----  
Process exited after 24.06 seconds with return value 0  
Pressione qualquer tecla para continuar. . .
```

# Relembrar Não Custa Nada.....

Diagrama de Blocos

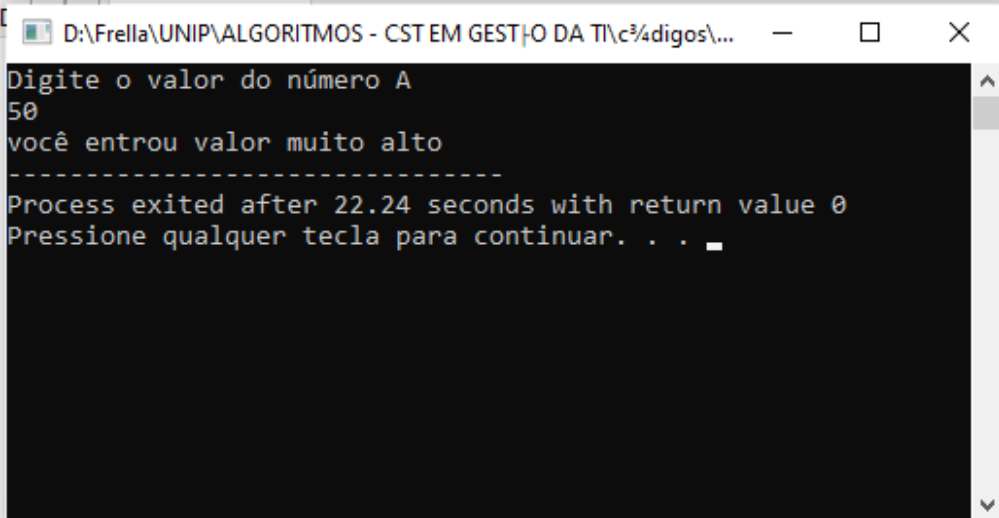


Português Estruturado

```
programa DECISÃO_SEQUENCIAL
var
  N : inteiro
início
  leia N
  se (N = 1) então
    escreva "você entrou o valor 1"
  fim_se
  se (N = 2) então
    escreva "você entrou o valor 2"
  fim_se
  se (N < 1) então
    escreva "você entrou valor muito baixo"
  fim_se
  se (N > 2) então
    escreva "você entrou valor muito alto"
  fim_se
fim
```

# Relembrar Não Custa Nada.....

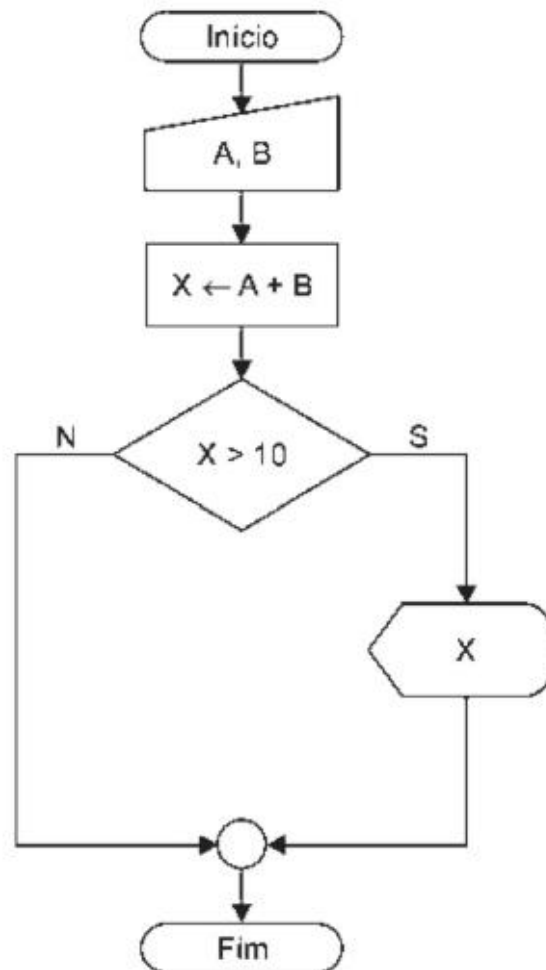
```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <locale.h> //necessário para usar setlocale
4
5  int main (void) {
6      //Utilizando caracteres e acentuação da língua portuguesa
7      setlocale(LC_ALL, "Portuguese");
8
9      int N;
10
11     printf ("Digite o valor do número A\n");
12     scanf ("%d",&N);
13
14     if (N == 1) {
15         printf ("você entrou o valor 1");
16     } else if (N == 2) {
17         printf ("você entrou o valor 2");
18     }
19     else if (N < 1){
20         printf ("você entrou valor muito baixo");
21     }else if (N > 2) {
22         printf ("você entrou valor muito alto");
23     }
24     return 0;
25 }
```



```
D:\Frella\UNIP\ALGORITMOS - CST EM GESTÃO DA TI\c%4digos\...
Digite o valor do número A
50
você entrou valor muito alto
-----
Process exited after 22.24 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```



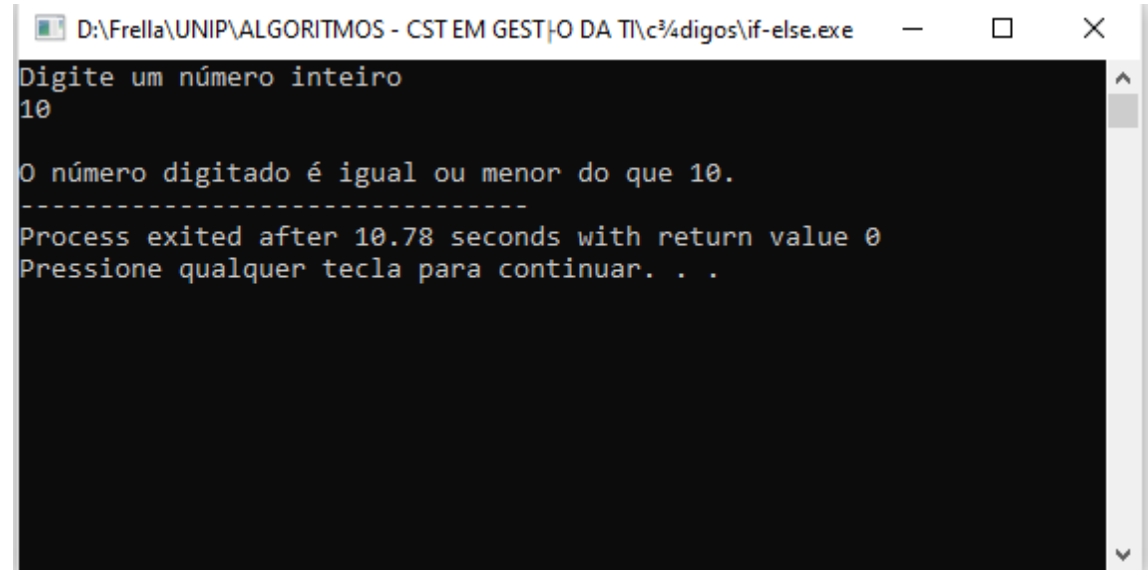
# Relembrar Não Custa Nada.....



```
programa SOMA_NÚMEROS
var
    X, A, B : inteiro
início
    leia A
    leia B
    X ← A + B
    se (X > 10) então
        escreva X
    fim_se
fim
```

# Relembrar Não Custa Nada.....

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <locale.h> //necessário para usar setlocale
4
5  /* ----- Estrutura If...else ----- */
6  int main (void) {
7
8      //Utilizando caracteres e acentuação da língua portuguesa
9      setlocale(LC_ALL, "Portuguese");
10
11     int numero;
12
13     printf ("Digite um número inteiro\n");
14     scanf ("%d",&numero);
15
16     if (numero>10){
17         printf("\nO número digitado é maior do que 10.");
18     } else {
19         printf("\nO número digitado é igual ou menor do que 10.");
20     }
21
22     return 0;
23 }
24
```



```
D:\Frella\UNIP\ALGORITMOS - CST EM GESTÃO DA TI\c34digos\if-else.exe
Digite um número inteiro
10

O número digitado é igual ou menor do que 10.
-----
Process exited after 10.78 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

# TESTE DA APROVAÇÃO DE UM ALUNO

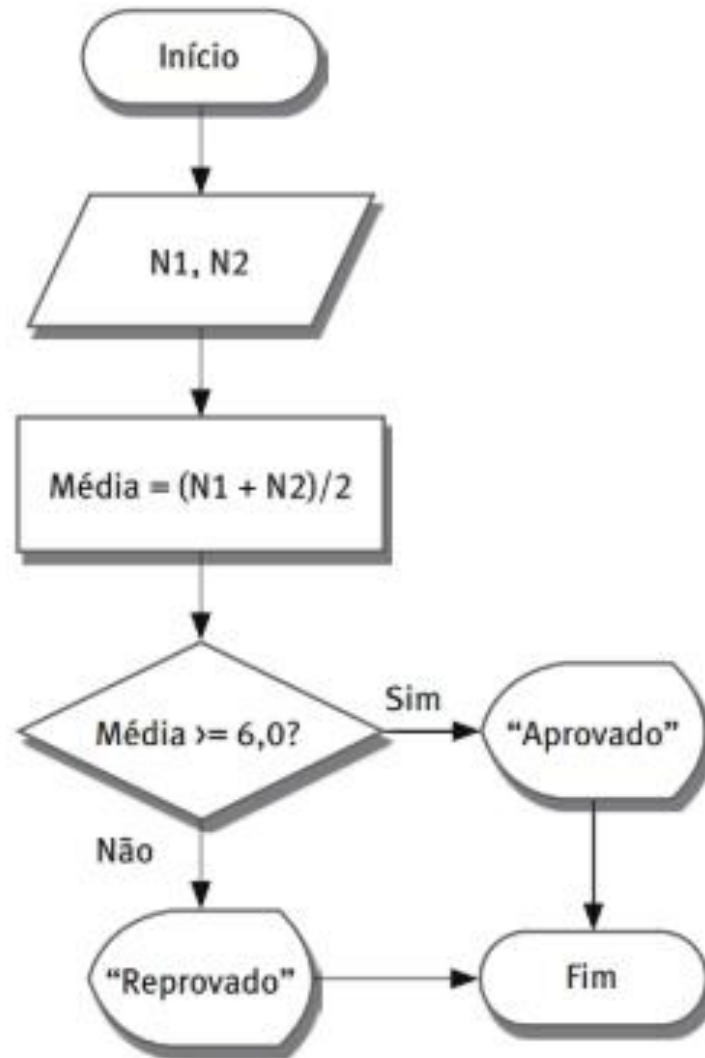
**Dados de entrada:** notas bimestrais.

**Processamento:** cálculo da média final e teste em relação à média de aprovação.

**Dados de saída:** “aprovado” ou “reprovado”.

O algoritmo receberá duas notas de avaliações, calculará a média aritmética dessas notas e comparará a média com o valor 6,0 (que é a média de aprovação da escola). Agora vem a decisão: Se a média for maior ou igual a 6,0 (seis), o aluno está “aprovado”; caso contrário, ele está “reprovado”

# TESTE DA APROVAÇÃO DE UM ALUNO



# TESTE DA APROVAÇÃO DE UM ALUNO

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <locale.h> //necessário para usar setlocale
4  /* ----- Estrutura If...else ----- */
5  int main (void) {
6      //Utilizando caracteres e acentuação da língua portuguesa
7      setlocale(LC_ALL, "Portuguese");
8
9      float nota1, nota2, mediaNotas;
10
11     printf ("Digite o valor da primeira nota\n");
12     scanf ("%f",&nota1);
13
14     printf ("Digite o valor da segunda nota\n");
15     scanf ("%f",&nota2);
16
17     mediaNotas = (nota1 + nota2) / 2;
18
19     if (mediaNotas >= 6){
20         printf("\nAluno aprovado.");
21     } else {
22         printf("\nAluno reprovado.");
23     }
24     return 0;
25 }
26
```

D:\Frella\UNIP\ALGORITMOS - CST EM GESTÃO DA TI\c34digos\4.5.1.2 If...else.exe

Digite o valor da primeira nota

5

Digite o valor da segunda nota

10

Aluno aprovado.

Process exited after 37.82 seconds with return value 0

Pressione qualquer tecla para continuar. . .

# TESTE DE CONHECIMENTO

Desenvolver um programa que solicite a entrada de um valor numérico inteiro e apresente uma das seguintes mensagens: "você entrou o valor 1" se for dada a entrada do valor numérico 1; "você entrou o valor 2" se for dada a entrada do valor numérico 2; "você entrou valor muito baixo" se for dada a entrada de um valor numérico menor que 1 ou "você entrou valor muito alto" se for dada a entrada de um valor numérico maior que 2.

Veja a descrição das etapas de entendimento do problema e a representação das ações a serem efetuadas pelo programa:

- a. Definir a entrada de um valor numérico inteiro (variável N).
- b. Verificar se  $N = 1$  e se for, apresentar a mensagem "você entrou o valor 1".
- c. Verificar se  $N = 2$  e se for, apresentar a mensagem "você entrou o valor 2".
- d. Verificar se  $N < 1$  e se for, apresentar a mensagem "você entrou valor muito baixo".
- e. Verificar se  $N > 2$  e se for, apresentar a mensagem "você entrou valor muito alto"

# UTILIZANDO NOSSO RASCUNHO.....

Diagrama de Blocos



Português Estruturado

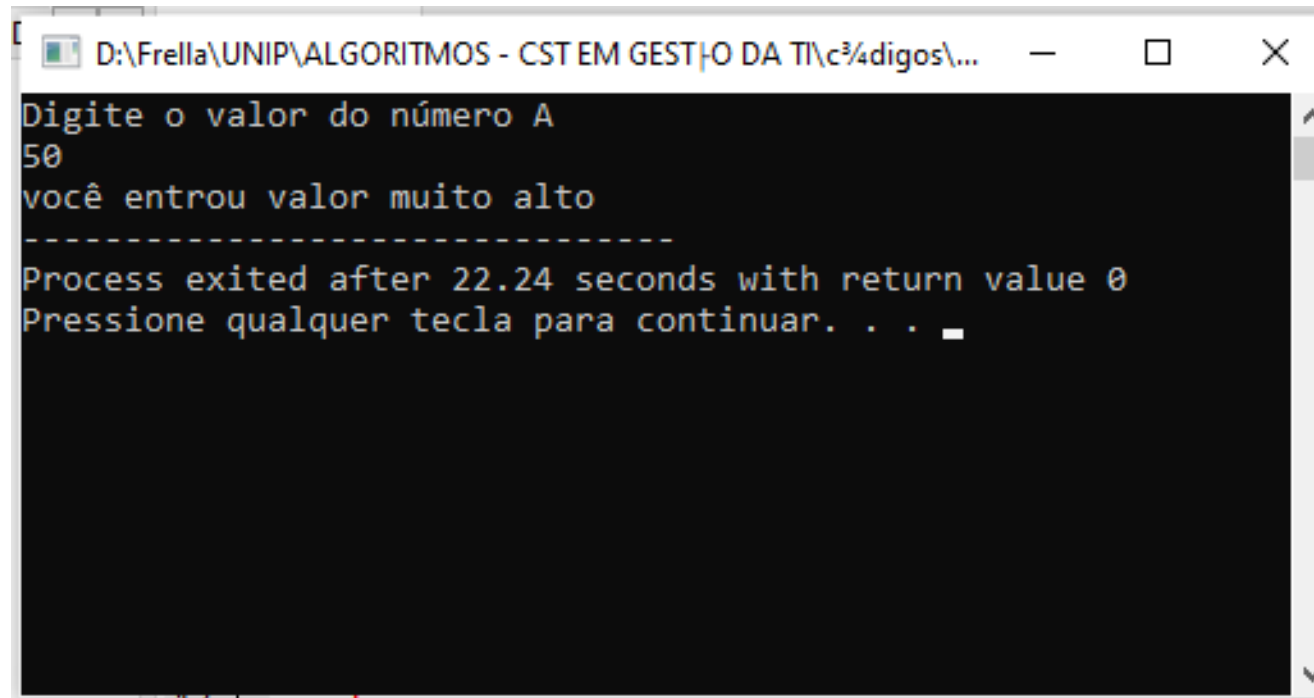
```
programa DECISÃO_SEQUENCIAL
var
  N : inteiro
início
  leia N
  se (N = 1) então
    escreva "você entrou o valor 1"
  fim_se
  se (N = 2) então
    escreva "você entrou o valor 2"
  fim_se
  se (N < 1) então
    escreva "você entrou valor muito baixo"
  fim_se
  se (N > 2) então
    escreva "você entrou valor muito alto"
  fim_se
fim
```

# IMPLEMENTANDO UTILIZANDO A LINGUAGEM C

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <locale.h> //necessário para usar setlocale
4
5  int main (void) {
6      //Utilizando caracteres e acentuação da língua portuguesa
7      setlocale(LC_ALL, "Portuguese");
8
9      int N;
10
11     printf ("Digite o valor do número A\n");
12     scanf ("%d",&N);
13
14     if (N == 1) {
15         printf ("você entrou o valor 1");
16     } else if (N == 2) {
17         printf ("você entrou o valor 2");
18     }
19     else if (N < 1){
20         printf ("você entrou valor muito baixo");
21     }else if (N > 2) {
22         printf ("você entrou valor muito alto");
23     }
24     return 0;
25 }
```



# IMPLEMENTANDO UTILIZANDO A LINGUAGEM C

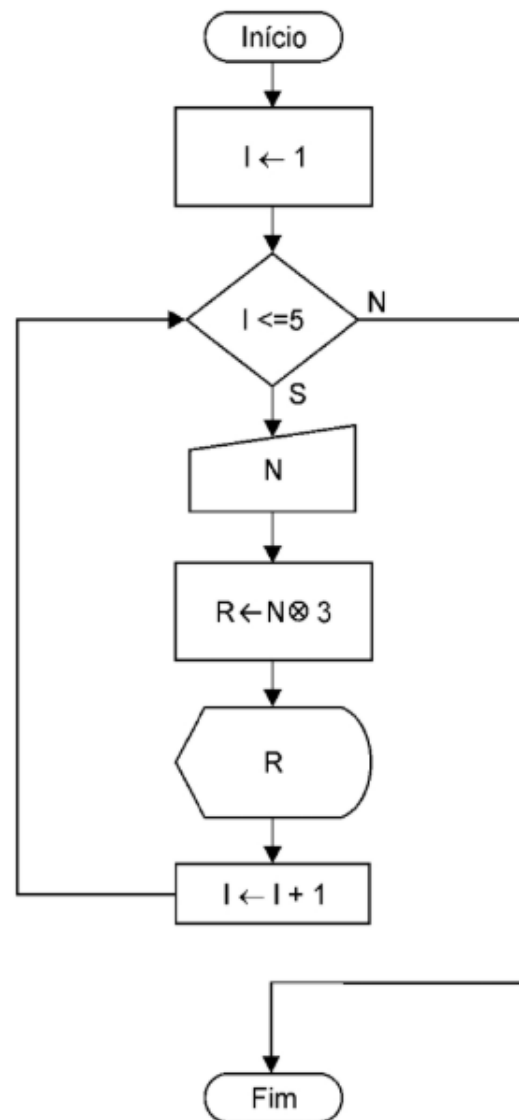


```
D:\Frella\UNIP\ALGORITMOS - CST EM GESTÃO DA TI\c³/4digos\...
Digite o valor do número A
50
você entrou valor muito alto
-----
Process exited after 22.24 seconds with return value 0
Pressione qualquer tecla para continuar. . . .
```

## RELEMBRAR UM POUCO MAIS....

Elaborar um programa que efetue a entrada de um valor numérico inteiro qualquer. Em seguida, processar o cálculo do valor de entrada, multiplicando-o por 3 e apresentando seu resultado. Proceder à execução dos passos anteriores cinco vezes.

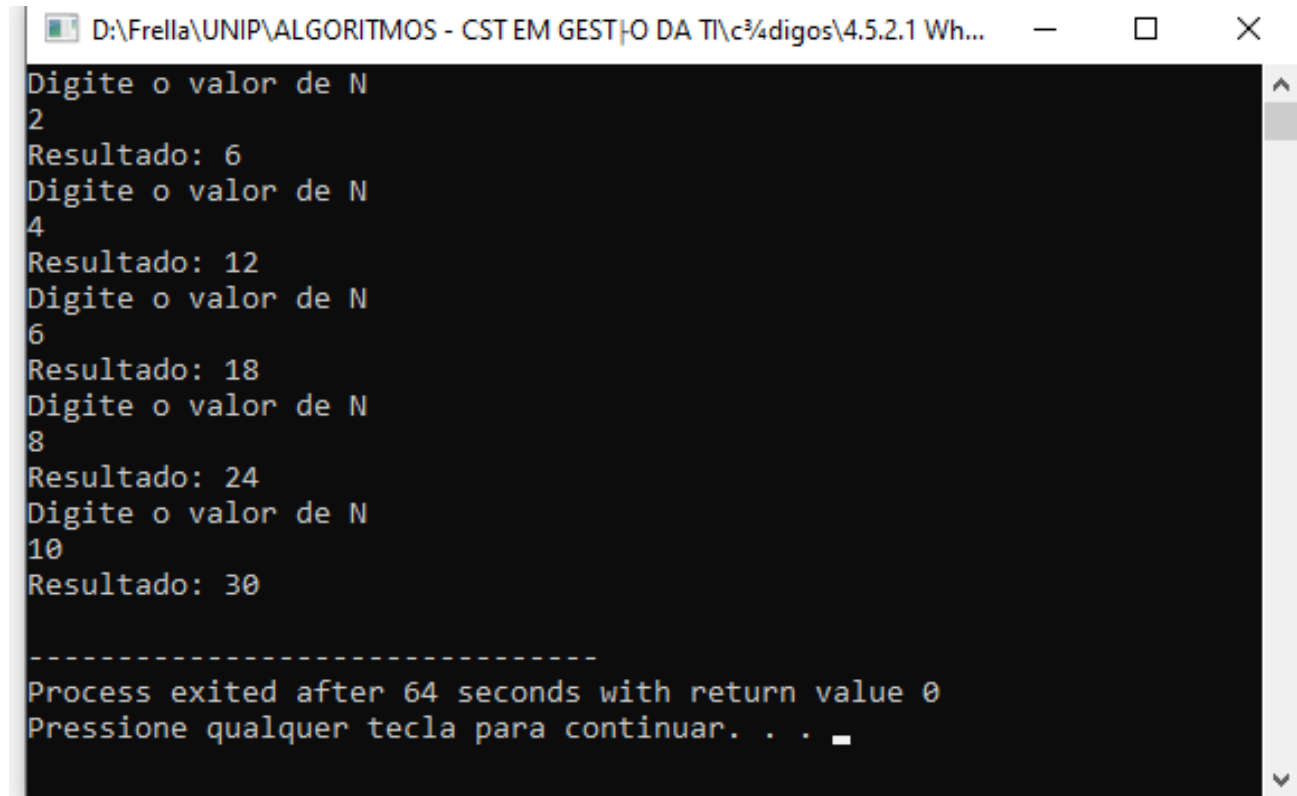
# RELEMBRAR UM POUCO MAIS....



```
programa LAÇO_PRÉ_TESTE_VERDADEIRO_VA
var
  I, N, R : inteiro
início
  I ← 1
  enquanto (I ≤ 5) faça
    leia N
    R ← N * 3
    escreva R
    I ← I + 1
  fim_enquanto
fim
```

# RELEMBRAR UM POUCO MAIS....

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main (void) {
5
6      int I, N, R;
7      I = 1;
8
9      while(I <=5){
10
11          printf ("Digite o valor de N\n");
12          scanf ("%d",&N);
13
14          R = N * 3;
15
16          printf("Resultado: %d\n", R);
17          I = I + 1;
18      }
19      return 0;
20  }
```



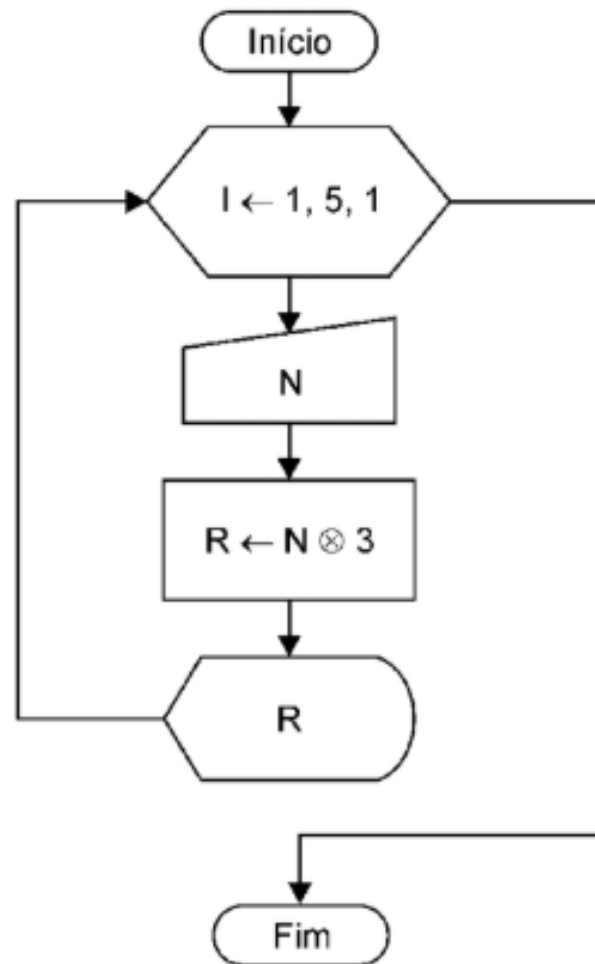
```
D:\Frella\UNIP\ALGORITMOS - CST EM GESTÃO DA TI\c34digos\4.5.2.1 Wh...
Digite o valor de N
2
Resultado: 6
Digite o valor de N
4
Resultado: 12
Digite o valor de N
6
Resultado: 18
Digite o valor de N
8
Resultado: 24
Digite o valor de N
10
Resultado: 30

-----
Process exited after 64 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

## RELEMBRAR UM POUCO MAIS....

A título de ilustração de uso do laço incondicional em um contexto operacional, considere o problema a seguir, observando detalhadamente as etapas de ação de um programador de computador: entendimento, diagramação e codificação. Elaborar um programa que efetue a entrada de um valor numérico inteiro qualquer. Em, seguida, calcular o valor de entrada, multiplicando-o por 3 e apresentando seu resultado. Proceder à execução dos passos anteriores cinco vezes.

# RELEMBRAR UM POUCO MAIS....



```
programa LAÇO_INCONDICIONAL
var
    I, N, R : inteiro
início
    para I de 1 até 5 passo 1 faça
        leia N
        R ← N * 3
        escreva R
    fim_para
fim
```

## RELEMBRAR UM POUCO MAIS....

O problema exposto já é conhecido, mas desta vez sua operação será mais dinâmica. Atente para os passos descritos a seguir:

1. Definir uma variável do tipo contador (variável I), variando de 1 a 5, de 1 em 1.
2. Ler um valor inteiro qualquer (variável N).
3. Efetuar a multiplicação do valor de N por 3, colocando o resultado na variável R.
4. Apresentar o valor calculado que está armazenado na variável R.
5. Repetir os passos 2, 3, 4 e 5 até o valor da variável chegar a 5.
6. Encerrar a execução do programa.

## RELEMBRAR UM POUCO MAIS....

O conjunto de instruções subordinadas ao bloco adjacente é executado entre os comandos para e fim\_para, sendo a variável I (variável de controle) inicializada com valor 1 e incrementada de mais 1 por meio do comando passo até a variável I atingir o valor 5. Esse tipo de estrutura pode ser utilizado todas as vezes que houver a necessidade de repetir trechos finitos, em que se conhecem os valores do laço: inicial, final e incremento.



# RELEMBRAR UM POUCO MAIS....

```
1  #include <stdio.h>
2
3  int main (void) {
4      int I, N, R;
5      char RESP;
6
7      for (I = 1; I<=5; I++){
8          printf ("Digite o valor de N\n");
9          scanf ("%d",&N);
10
11          R = N * 3;
12
13          printf("Resultado: %d\n", R);
14      }
15      return 0;
16  }
```

D:\Frella\UNIP\ALGORITMOS - CST EM GESTÃO DA TI\

```
Digite o valor de N
1
Resultado: 3
Digite o valor de N
2
Resultado: 6
Digite o valor de N
3
Resultado: 9
Digite o valor de N
4
Resultado: 12
Digite o valor de N
5
```

# BIBLIOTECAS

**stdio.h** - Essa biblioteca é a mais utilizada na programação em linguagem C, pois é a padrão, na qual estão embutidas as funções printf(), puts(), gets(), scanf(), entre outras.

**math.h** - Possui as funções matemáticas usadas pela linguagem. Encontram-se funções trigonométricas, hiperbólicas, exponenciais, logarítmicas, entre outras.

**string.h** - Esta possui as rotinas de tratamento de strings e caracteres, na qual se encontram as funções strcmp() e strcpy(), entre outras.

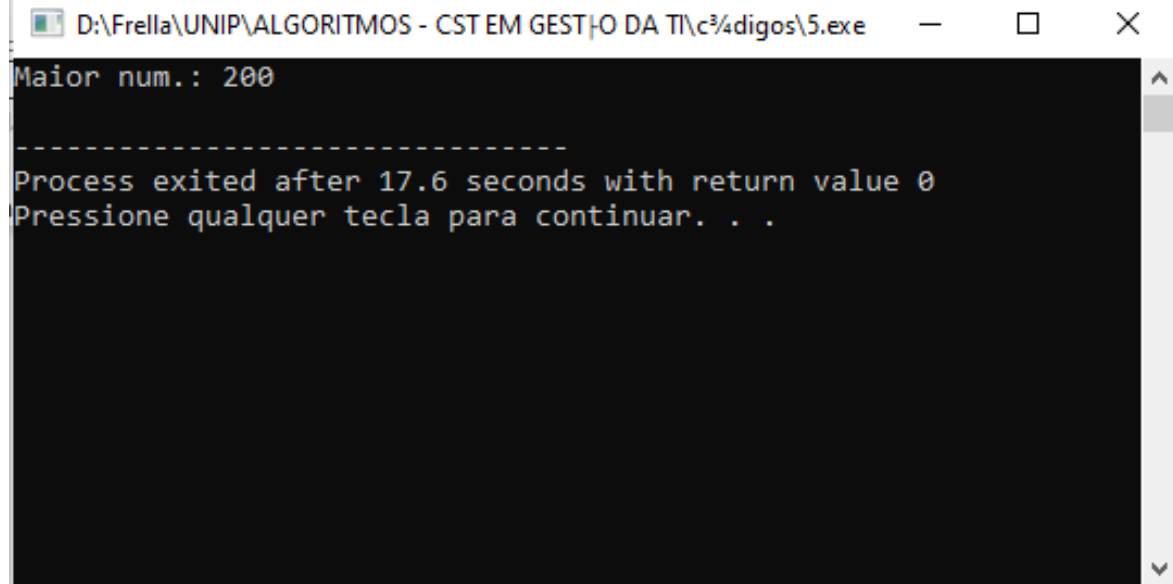
**time.h** - Essa biblioteca possui as funções de manipulação de data e hora do sistema.

**stdlib.h** - Possui um conjunto de funções que não se enquadra em outras categorias.

# BIBLIOTECAS

main.cpp write.cpp [\*] 5.cpp

```
1  #include <stdio.h>
2
3  /* declaração da função */
4  int max(int num1, int num2);
5  int main (){
6      /* variável local*/
7      int a = 100;
8      int b = 200;
9      int ret;
10
11     /* chamada da função */
12     ret = max(a, b);
13     printf( "Maior num.: %d\n", ret );
14     return 0;
15 }
16
17 /* function*/
18 int max(int num1, int num2){
19     int result;
20     if (num1 > num2)
21         result = num1;
22     else
23         result = num2;
24     return result;
25 }
```



```
D:\Frella\UNIP\ALGORITMOS - CST EM GESTÃO DA TI\c34digos\5.exe
Maior num.: 200
-----
Process exited after 17.6 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

# FORMATAÇÃO DE IMPRESSÃO

Toda ação de entrada e saída efetivada com a linguagem C, independentemente de essas ações se darem de forma direta ou indireta, sempre realiza a ação do tratamento de cadeias ou caracteres e sua conversão para o tipo adequado usado pela linguagem.

Em diversos momentos da apresentação desta obra foram usadas para as operações de entrada e saída de dados diversas funções, tais como as funções de entrada `scanf()` / `scanf_s()`, `getchar()`, `fgetc()`, `fgets()`, `fscanf()` / `fscanf_s()` e `sscanf()` / `sscanf_s()`, além das funções de saída `printf()` / `printf_s()`, `putchar()`, `fputc()`, `fputs()`, `fprintf()` / `fprintf_s()`, `puts()` e `sprintf()` / `sprintf_s()`, `snprintf()` / `snprintf_s()`.

As ações foram efetivadas de diversas maneiras, desde por meio dos periféricos-padrão de entrada (teclado) e saída (monitor de vídeo) até operações com arquivos em disco e buffers de memória, mas sempre contextualizadas à medida que eram necessárias.

# FORMATAÇÃO DE IMPRESSÃO

- Gravar e ler um caractere por vez: `fputc()` e `fgetc()`
- Ler e gravar linha a linha: `fputs()` e `fgets()`
- Ler e gravar dados formatados: `fprintf()` e `fscanf()`
- Ler e gravar blocos de bytes: `fwrite()` e `fread()`

# FORMATAÇÃO DE IMPRESSÃO

- Gravar e ler um caractere por vez: fputc()

```
#include <stdio.h>
int main(int argc, char*argv[])
{
    FILE *f = fopen("teste.txt", "w");
    char i;
    for (i='a'; i<='z'; i++)
        fputc(i, f);
    fclose(f);
}
```

# FORMATAÇÃO DE IMPRESSÃO

- Gravar e ler um caractere por vez: fgetc()

```
#include <stdio.h>
int main(int argc, char *argv[])
{
    FILE *f = fopen("teste.txt", "r");
    char i;
    while((i=fgetc(f)) != EOF)
        printf("%c", i);
    fclose(f);
}
```

# FORMATAÇÃO DE IMPRESSÃO

- Validar se o arquivo existe

```
#include <stdio.h>
int main(int argc, char *argv[])
{
    FILE *f; char i;
    if((f = fopen("teste2.txt", "r")) == NULL)
    {
        printf("Nao foi possivel abrir o arquivo\n");
        return 0;
    }

    while((i=fgetc(f)) != EOF)
        printf("%c", i);
    fclose(f);
    return 0;
}
```



# FORMATAÇÃO DE IMPRESSÃO

- Ler e gravar linha a linha: fputs()

```
#include <stdio.h>
int main(int argc, char *argv[])
{
    FILE *f = fopen("testeSTR.txt", "w");
    fputs("Desta maneira, a complexidade dos \n", f);
    fputs("estudos efetuados estimula a \n", f);
    fputs("padronização das diversas correntes de pensamento.", f);
    fclose(f);
    return 0;
}
```

# FORMATAÇÃO DE IMPRESSÃO

- Exemplo: Imprimir os números de 1 até 10 no arquivo teste.txt

```
#include <stdio.h>
int main(int argc, char *argv[])
{
    FILE *f = fopen("teste.txt", "w");
    int i;
    for (i=1; i<=10; i++)
        fprintf(f, "%d\n", i);
    fclose(f);
}
```

# FORMATAÇÃO DE IMPRESSÃO

- Exemplo: Imprimir os números que estão no arquivo teste.txt

```
#include <stdio.h>
int main(int argc, char *argv[])
{
    FILE *f = fopen("teste.txt", "r");
    int i;
    while (fscanf(f, "%d", &i) == 1)
        printf("%d\n", i);
    fclose(f);
}
```

# FORMATAÇÃO DE IMPRESSÃO

- Exemplo: Conteúdo do arquivo teste.txt: joao maria jose

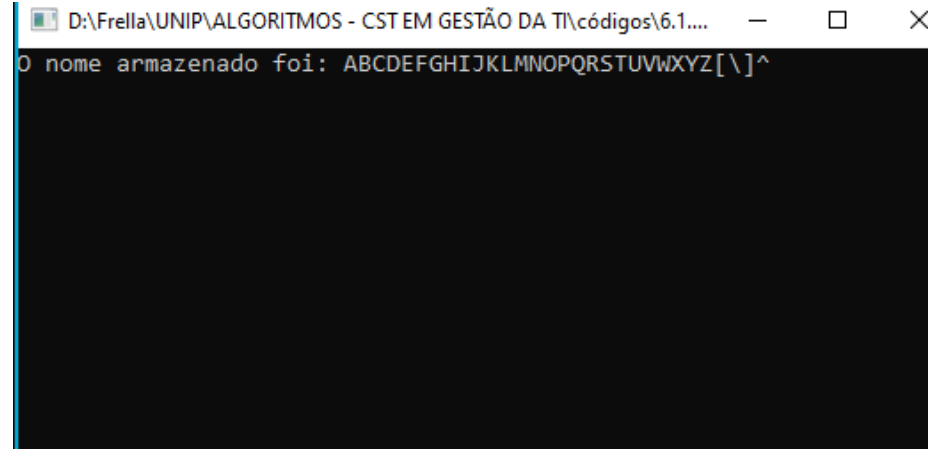
```
#include <stdio.h>
int main(int argc, char *argv[])
{
    FILE *f = fopen("teste.txt", "r");
    char palavra[100];
    while (fscanf(f, "%s", palavra) == 1)
        printf("%s\n", palavra);
    fclose(f);
}
```

# FORMATAÇÃO DE IMPRESSÃO

A linguagem C, não determina um tipo específico para a manipulação de strings (que são vetores ou cadeias de caracteres, terminados pelo caractere NULL). Para isso, fornece uma completa biblioteca de funções específicas (string.h). Funções que manipulam strings a percorrem até encontrar o caractere NULL, quando saberão que ela terminou. Utilizamos, portanto, o caractere zero da tabela ASCII ('\0') para encerrar a string, e este ocupa um espaço que deve ser previsto pelo programador, como visto anteriormente. A relação entre strings e vetores é, dessa forma, direta. Uma string é um vetor de caracteres, mas nem todo vetor de caracteres é uma string.

# FORMATAÇÃO DE IMPRESSÃO

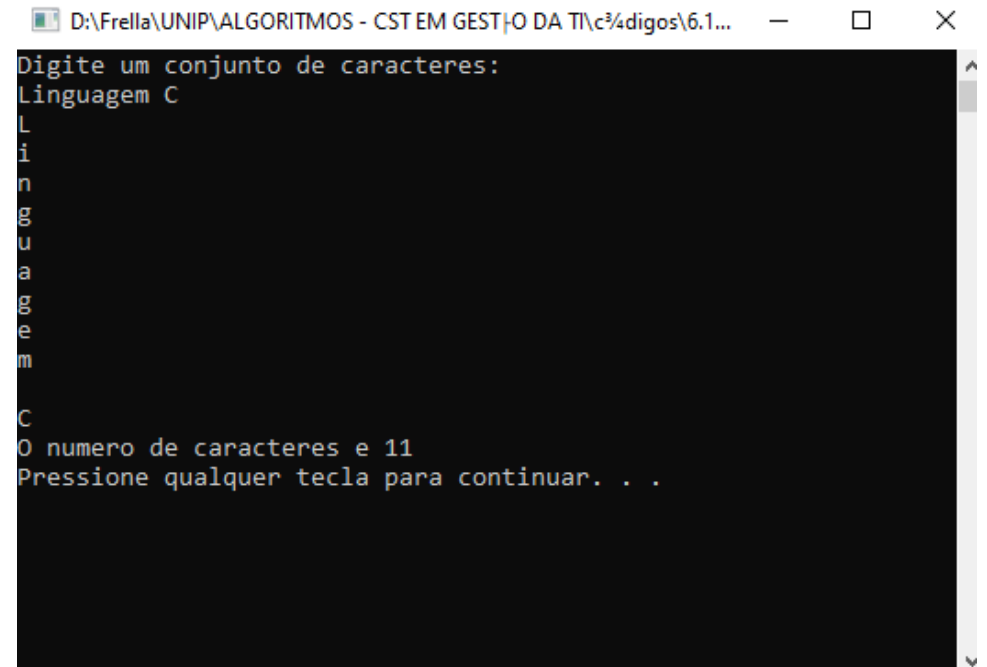
```
1  #include<stdio.h>
2  #include<conio.h>
3
4  int main(void){
5      char frase[50];
6
7
8      int i;
9      for(i=0; i < 30; i++) {
10         frase[i] = 'A' + i; /* a variável 'i' incrementa a posição do caractere na Tabela ASCII */
11     }
12
13     printf("O nome armazenado foi: %s", frase);
14
15     getch();
16     return 0;
17 }
18
```



The screenshot shows a Windows command prompt window with the title bar "D:\Frella\UNIP\ALGORITMOS - CST EM GESTÃO DA TI\códigos\6.1....". The window contains the output of the C program: "O nome armazenado foi: ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^". The text is displayed in a monospaced font on a black background.

# FORMATAÇÃO DE IMPRESSÃO

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  main() {
5      //Declaração das variáveis
6      char string[50];
7      int i;
8
9      //Entrada de Dados
10     printf("Digite um conjunto de caracteres: \n");
11     gets(string);
12
13     //Laço de repetição - FOR
14     for(i=0; string[i] != NULL; i++){
15         putchar(string[i]); printf("\n");
16     }
17
18     //Saída de Dados
19     printf("O numero de caracteres e %d \n", i);
20     system("pause");
21 }
22
```



```
D:\Frella\UNIP\ALGORITMOS - CST EM GESTÃO DA TI\c\4digos\6.1...
Digite um conjunto de caracteres:
Linguagem C
L
i
n
g
u
a
g
e
m
C
O numero de caracteres e 11
Pressione qualquer tecla para continuar. . .
```

# FORMATAÇÃO DE IMPRESSÃO

Outras importantes funções de manipulação de strings são:

- **strcpy( )**: copia uma string em outra;
- **strcat( )**: adiciona o conteúdo de uma string em outra;
- **strlwr( )**: converte conteúdo para minúsculas;
- **strupr( )**: converte conteúdo para maiúsculas;
- **strcmp( )**: compara duas strings.



# FORMATAÇÃO DE IMPRESSÃO

No exemplo abaixo é possível verificar a aplicação da função `strcmp( )` que realiza a comparação de duas String que serão digitadas pelo usuário.

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <stdlib.h>
4
5  main() {
6      // Declaração das variáveis
7      char x[10], y[10];
8
9      //Entrada de Dados
10     printf("Entre a primeira string: \n");
11     gets(x);
12     printf("Entre a segunda string: \n");
13     gets(y);
14
15     //Comparação
16     if(strcmp(x, y) == 0){
17         printf("Iguais! \n");
18     }else{
19         printf("Diferentes! \n");
20     }
21
22     system("pause");
23 }
```

D:\Frella\UNIP\ALGORITMOS - CST EM GESTÃO DA TI\c34digos\6.1.1 Entrad...

```
Entre a primeira string:
Linguagem C
Entre a segunda string:
Linguagem Java
Diferentes!
Pressione qualquer tecla para continuar. . .
```

# ARQUIVOS

A principal vantagem de um arquivo é que as informações armazenadas podem ser consultadas a qualquer momento. Outra vantagem é o fato de armazenar um número maior de registros do que uma tabela em memória. Está limitado apenas ao tamanho do meio físico para gravação. É um conjunto de registros (que pode ser apenas um registro) que, por sua vez, é um conjunto de campos (que pode ser apenas um campo), sendo cada campo o conjunto de informações.

# ARQUIVOS

De acordo com MANZANO (2013), o tipo de abertura de um arquivo é especificado por três códigos do tipo string, a saber: letra r para leitura (read), letra w para gravação (write) e letra a para adicionar dados (append), segue a tabela com essa representação:

Tipo de Abertura	Descrição
R	Este código permite apenas abrir um arquivo texto para leitura de seus dados. É necessário que o arquivo esteja presente no disco.
W	Este código permite apenas abrir um arquivo texto para escrita (gravação). Este código cria o arquivo para ser trabalhado. Caso o arquivo exista, este código recria o arquivo, ou seja, você perde o arquivo criado anteriormente. Deve ser usado com muito cuidado.
A	Este código permite apenas abrir um arquivo texto para escrita (gravação), permitindo acrescentar novos dados ao final dele. Caso o arquivo não exista, ele será então criado.

# ARQUIVOS

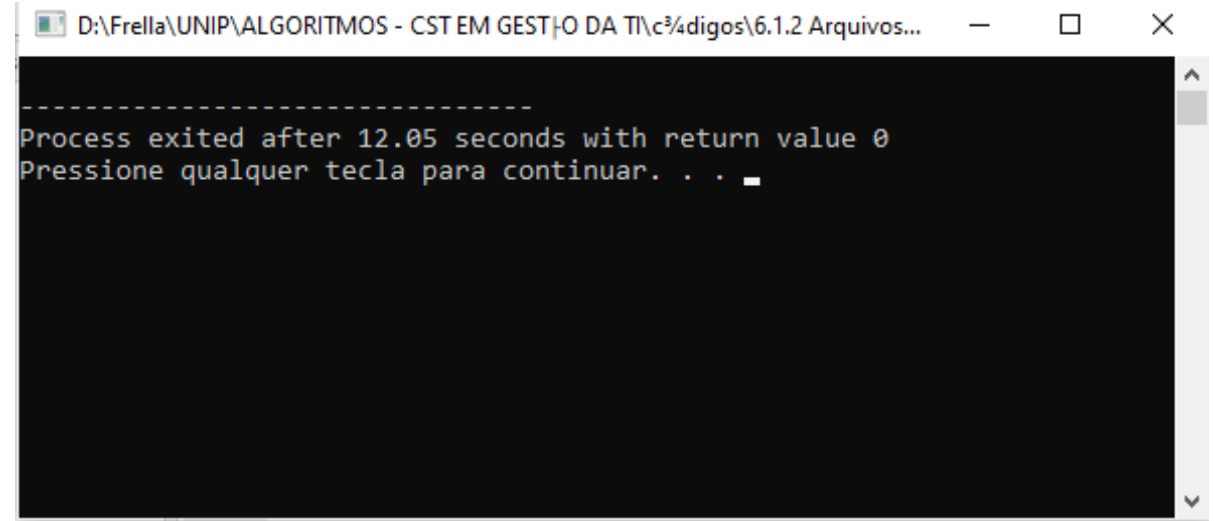
Na tabela abaixo são apresentadas as funções específicas para se trabalhar com arquivos:

Funções de manipulação de arquivos	
<b>fopen( )</b>	abre um arquivo para trabalho
<b>fclose( )</b>	fecha o arquivo
<b>putc( )</b>	escreve um caractere no arquivo aberto
<b>fputc( )</b>	mesma função de putc( )
<b>getc( )</b>	lê um caractere do arquivo de trabalho
<b>fgetc( )</b>	mesma função de getc( )
<b>fseek( )</b>	posiciona o ponteiro de arquivo em um byte específico
<b>rewind( )</b>	posiciona o ponteiro de arquivo no início deste
<b>fprintf( )</b>	idem ao printf na saída-padrão
<b>fscanf( )</b>	idem ao scanf na entrada-padrão

# ARQUIVOS

Para iniciar as operações com arquivo é necessário criá-lo. No exemplo abaixo será apresentado o código para a criação do arquivo e apresentado o print da criação após a sua execução.

```
2.cpp 1.cpp
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(void) {
5
6      // Definicao do ponteiro para o arquivo
7      FILE *PTRARQ;
8
9      //Criação do Arquivo
10     PTRARQ = fopen("ARQTXT01.txt", "a");
11
12     //Finalizar o Arquivo
13     fclose(PTRARQ);
14
15 }
16
```



The screenshot shows a Windows command prompt window with the title bar "D:\Frella\UNIP\ALGORITMOS - CST EM GESTÃO DA TI\c34digos\6.1.2 Arquivos...". The window contains the following text:

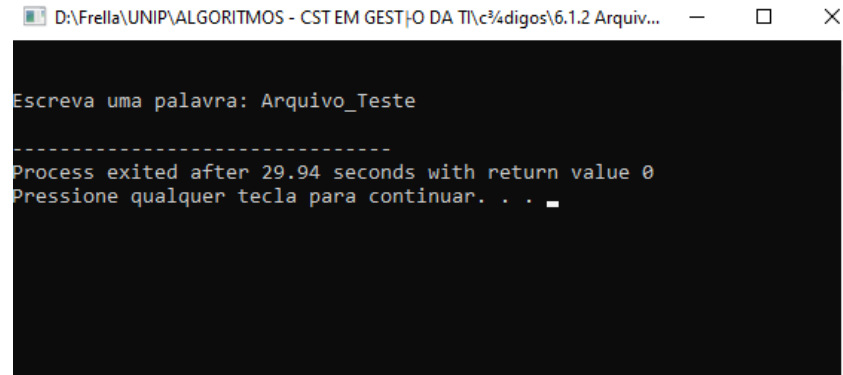
```
-----
Process exited after 12.05 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

# ARQUIVOS

2.cpp

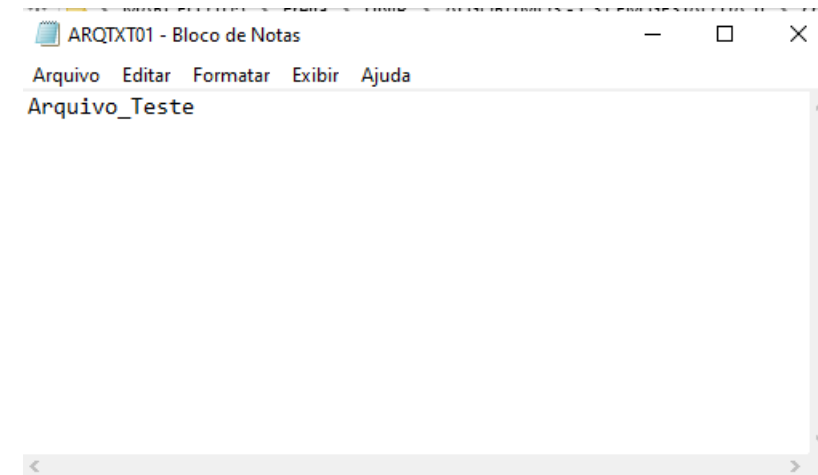
1.cpp

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(void) {
5      //Declaração das variáveis
6      FILE *PTRARQ;
7      char PALAVRA[20];
8
9      //Acessar o arquivo gerado no código anterior
10     PTRARQ = fopen("ARQTXT01.txt", "w");
11
12     // Solicitar ao usuário uma palavra
13     printf("\n\nEscreva uma palavra: ");
14     scanf("%s", PALAVRA);
15
16     //Inserir a palavra digitada no arquivo
17     fprintf(PTRARQ, "%s", PALAVRA);
18
19     //Fechar o arquivo
20     fclose(PTRARQ);
21
22     return 0;
23 }
24
```



D:\Frella\UNIP\ALGORITMOS - CST EM GESTÃO DA TI\c\4digos\6.1.2 Arquiv... — □ ×

```
Escreva uma palavra: Arquivo_Testes
-----
Process exited after 29.94 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```



ARQTXT01 - Bloco de Notas — □ ×

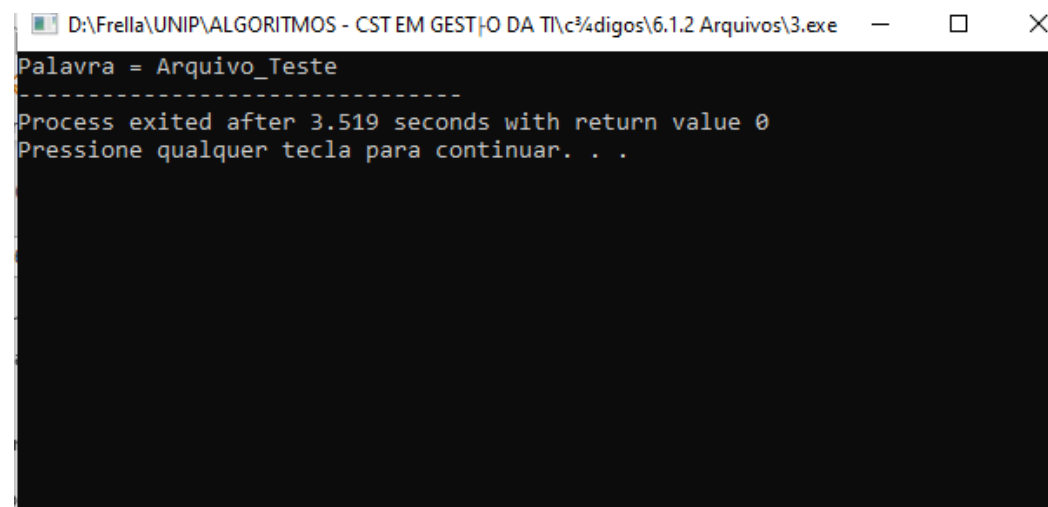
Arquivo Editar Formatar Exibir Ajuda

Arquivo\_Testes

# ARQUIVOS

O próximo exemplo será possível acessar as informações que foram escritas no arquivo texto, exibindo o seu conteúdo. Segue o código com essa ação:

```
3.cpp  1.cpp
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(void) {
5      // Declaração das variáveis
6      FILE *PTRARQ;
7      char PALAVRA[20];
8
9      // Leitura do Arquivo
10     PTRARQ = fopen("ARQTXT01.txt", "r");
11     fscanf(PTRARQ, "%s", PALAVRA);
12
13     // Exibição da palavra (do Arquivo)
14     printf("Palavra = %s", PALAVRA);
15
16     // Fechamento do Arquivo
17     fclose(PTRARQ);
18
19     return 0;
20 }
21
```

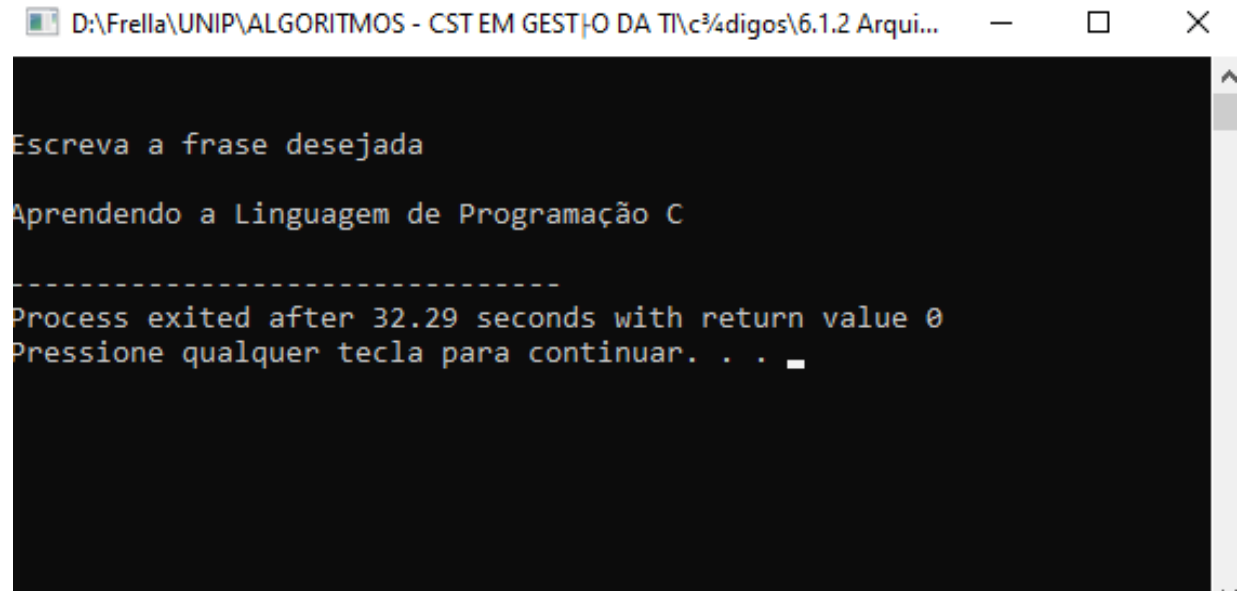


```
D:\Frela\UNIP\ALGORITMOS - CSTEM GEST|O DA TI\c%4digos\6.1.2 Arquivos\3.exe
Palavra = Arquivo_Teste
-----
Process exited after 3.519 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

# ARQUIVOS

No próximo exemplo é possível salvar o texto de uma frase no Arquivo que será gerado.

```
4.cpp 1.cpp
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(void) {
5      // Declaração das variáveis
6      FILE *PTRARQ;
7      char LETRA;
8
9      // Criação do Arquivo
10     PTRARQ = fopen("FRASE.txt", "w");
11
12     // Entrada de Dados - Solicitação da Frase
13     printf("\n\nEscreva a frase desejada\n\n");
14     while((LETRA = getchar()) != '\n')
15         putc(LETRA, PTRARQ);
16
17     // Fechamento do Arquivo
18     fclose(PTRARQ);
19
20     return 0;
21 }
22
```



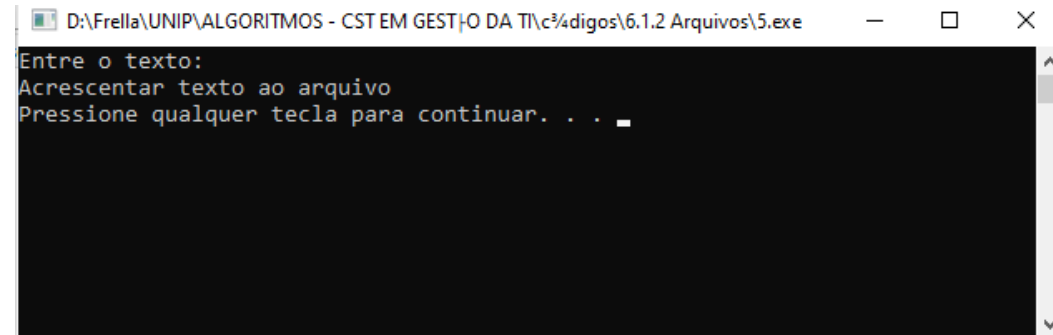
```
D:\Frella\UNIP\ALGORITMOS - CST EM GESTÃO DA TI\c34digos\6.1.2 Arqui...
Escreva a frase desejada
Aprendendo a Linguagem de Programação C
-----
Process exited after 32.29 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```



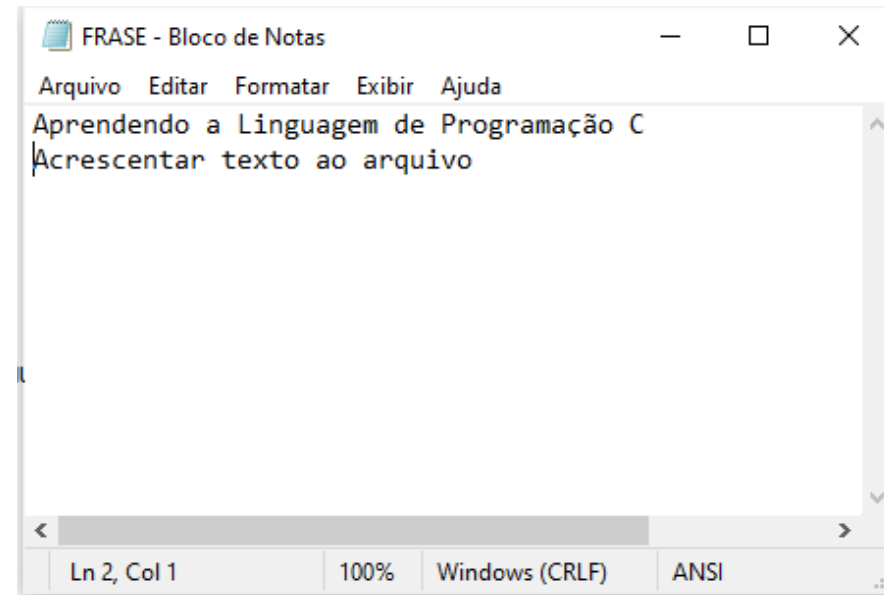
# ARQUIVOS

No próximo exemplo é possível abrir o arquivo em modo que permita acrescentar um novo texto ao conteúdo, e depois gravar a alteração.

```
5.cpp 1.cpp
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  main() {
5      // Declaração de variáveis
6      FILE * arquivo;
7      char mensagem[80];
8
9      // Entrada de Dados
10     printf("Entre o texto: \n");
11     gets(mensagem);
12
13     // Verificar se o arquivo existe
14     if((arquivo = fopen("FRASE.txt","a")) == NULL) {
15         printf("Erro de abertura! \n");
16     } else {
17         // Gravar o texto no Arquivo
18         fprintf(arquivo, "%s \n", mensagem);
19
20         // Fechar o arquivo
21         fclose(arquivo);
22     }
23     system("pause");
24 }
```



```
D:\Frella\UNIP\ALGORITMOS - CST EM GESTÃO DA TI\c34digos\6.1.2 Arquivos\5.exe
Entre o texto:
Acrescentar texto ao arquivo
Pressione qualquer tecla para continuar. . . .
```



```
FRASE - Bloco de Notas
Arquivo  Editar  Formatar  Exibir  Ajuda
Aprendendo a Linguagem de Programação C
Acrescentar texto ao arquivo
Ln 2, Col 1  100%  Windows (CRLF)  ANSI
```

# EXERCÍCIO

“Uma função é um bloco de código de programa que pode ser usado diversas vezes em sua execução. O uso de funções permite que o programa fique mais legível, mais bem estruturado. Um programa em C consiste, no fundo, de várias funções colocadas juntas” (Mesquita, 1998 p. 8).

# EXERCÍCIO

Nesse sentido, assinale a alternativa que contém a definição correta:

- A) A função é a forma para unificar todas as tarefas que serão executadas em um único local.
- B) Para declarar uma função, devemos informar ao compilador simplesmente o seu nome.
- C) A função é um grupo de instruções que juntas executam uma tarefa.
- D) Não podemos definir os seguintes elementos como funções: `main()`, `printf()`, `scanf()`, `fgets()`, `puts()`, `strcmp()`, `strcpy()`.
- E) É considerada função externa a chamada biblioteca de funções, como, por exemplo, as bibliotecas-padrão `stdio.h` e `string.h`.

# EXERCÍCIO

Nesse sentido, assinale a alternativa que contém a definição correta:

A) A função é a forma para unificar todas as tarefas que serão executadas em um único local.

B) Para declarar uma função, devemos informar ao compilador simplesmente o seu nome.

**C) A função é um grupo de instruções que juntas executam uma tarefa.**

D) Não podemos definir os seguintes elementos como funções: `main()`, `printf()`, `scanf()`, `fgets()`, `puts()`, `strcmp()`, `strcpy()`.

E) É considerada função externa a chamada biblioteca de funções, como, por exemplo, as bibliotecas-padrão `stdio.h` e `string.h`.

# Referências

- ART, Susan; ELLISON, Robert; FEILER, Peter; EDITED, A.; FRITZSON, Peter. (1992). *Overview of Software Development Environments*. Disponível em: <https://www.ics.uci.edu/~andre/ics228s2006/dartellisonfeilerhabermann.pdf>. Acesso em: 25 abr. 2020.
- BEALE, E. M. L. 1970. Matrix Generators and Output Analyzers. In: H. W. Kuhn (ed.), *Proceedings of the Princeton Symposium on Mathematical Programming*, Princeton University Press, Princeton, NJ, p. 25-36.
- BROOKSHEAR, J. G. *Computer Science: An Overview*. Boston, Mass.: Pearson. 2009.
- CHARNTAWEEKHUN, Kanis; WANGSIRIPITAK, Somkiat. (2006). *Visual Programming using Flowchart*. Disponível em: 10.1109/ISCIT.2006.339940. Acesso em: 25 abr. 2020.
- CREEGAN, J. B. 1985. *DATAFORM*, a Model Management System. Ketron, Inc., Arlington, VA.

# Referências

- DIFFERENCE BETWEEN SOURCE CODE AND OBJECT CODE. 24/01/2018. Disponível em: <https://www.differencebetween.com/wp-content/uploads/2018/01/Difference-Between-Source-Code-and-Object-Code.pdf>. Acesso em: 25 abr. 2020.
- MANZANO, José Augusto N. G. *Algoritmos: lógica para desenvolvimento de programação de computadores*. 29. ed. São Paulo: Érica, 2019.
- SLONNEGER, Kenneth; KURTZ, Barry L. *Formal syntax and semantics of programming languages: a laboratory*. 1995. Disponível em: <https://www.mobt3ath.com/uplode/book/book-26246.pdf>. Acesso em: 25 abr. 2020.
- SOFFNER, R. *Algoritmos e Programação em Linguagem C*. São Paulo: Saraiva, 2013.
- SOICHER, Leonard; VIVALDI, Franco. *Algorithmic Mathematics*. University of London, 2004. Disponível em: <http://www.maths.qmul.ac.uk/~Isoicher/ambook.pdf>. Acesso em: 25 abr. 2020.

# Referências

- STALLINGS, W. *Computer Organization and Architecture, Designing for Performance*. Boston, Mass.: Pearson. 2010.
- UNIVERSITY OF CRETE. 2020. *Introduction to Data Types and Structures*. Disponível em: <https://www.csd.uoc.gr/~hy252/html/References/Introduction%20to%20Data%20Types%20and%20Structures.pdf>. Acesso em: 28 abr. 2020.
- WALIA, Ravi K. 2020. *ALGORITHM & FLOWCHART MANUAL for STUDENTS*. University of Horticulture & Forestry - Índia. Disponível em: <http://www.yspuniversity.ac.in/cic/algorithm-manual.pdf>. Acesso em: 26 abr. 2020.
- WIRTH, N. *Algorithms and Data Structures*. 1985. Disponível em: <https://inf.ethz.ch/personal/wirth/AD.pdf>. Acesso em: 28 abr. 2020.
- YANG, Kuo-pao. *Chapter 6 - Data Types*. 2017. Disponível em: <https://www2.southeastern.edu/Academics/Faculty/kyang/2017/Fall/CMPS401/ClassNotes/CMPS401ClassNotesChap06.pdf>. Acesso em: 27 abr. 2020.