

LINGUAGEM E TÉCNICAS DE PROGRAMAÇÃO

Professor: Fábio e Professora: Karhyne

AULA 04

Estruturas de repetição (laços)



ESTRUTURAS DE REPETIÇÃO

Estruturas de Repetição:

A estrutura de repetição é utilizada quando precisamos repetir partes do programa. Com a vantagem de minimizar o número de linhas de código, deixando-o mais legível.

No Portugol Studio, existem três estruturas de repetição, são elas:

Enquanto (while)

Faça enquanto (do while)

Para (for)

Estruturas de Repetição:

ENQUANTO

O bloco do comando enquanto é executado, até que a condição seja falsa.

É necessário inicializar a variável de **controle** antes da execução o bloco.

É necessário alterar o valor da variável de controle dentro do bloco de repetição, para que em um determinado valor faça com que a condição seja falsa para sair do bloco de repetição. Caso contrário o código poderá ficar em laços infinito.

EXEMPLO

EXEMPLO DE ESTRUTURA DE REPETIÇÃO ENQUANTO

Programa que executa um laço de repetição que imprime "*", até que a condição seja falsa.

```
1 programa
2 {
3     funcao inicio()
4     {
5         /*declarando e inicializando
6         a variável de controle*/
7         inteiro c=1
8
9         enquanto(c <=3 ){
10             escreva("*")
11             c++
12         }
13     }
14 }
```

Inicialização

Condição

Incremento

Exemplo em C

```
1 #include <stdio.h>
2
3 int main(){
4     int c = 1;
5
6     while(c <= 3){
7         printf("*");
8         c = c + 1;
9     }
10
11     return 0;
12 }
```

EXEMPLO – TESTE DE MESA

EXEMPLO DE ESTRUTURA DE REPETIÇÃO ENQUANTO

Programa que executa um laço de repetição que imprime “*”, até que a condição seja falsa.

```
programa
{
  funcao inicio()
  {
    inteiro c // variável de controle
    c = 1 // iniciando a variável de controle
    enquanto (c<=3)
    {
      escreva("*")
      c = c + 1;
    }
  }
}
```

Tabela 1 – Teste de Mesa

Variável	Condição	
c = 1	c <= 3 = true	*
c = c + 1 = 2	c <= 3 = true	*
c = c + 1 = 3	c <= 3 = true	*
c = c + 1 = 4	c <= 3 = false	

Estruturas de Repetição:

FAÇA ENQUANTO

O bloco do comando para enquanto é executado, até que a condição seja falsa.

Exemplo:

condicao = 4

faca {

bloco de instruções

condicao++

} enquanto (condicao < 10)

Inicialização

Incremento

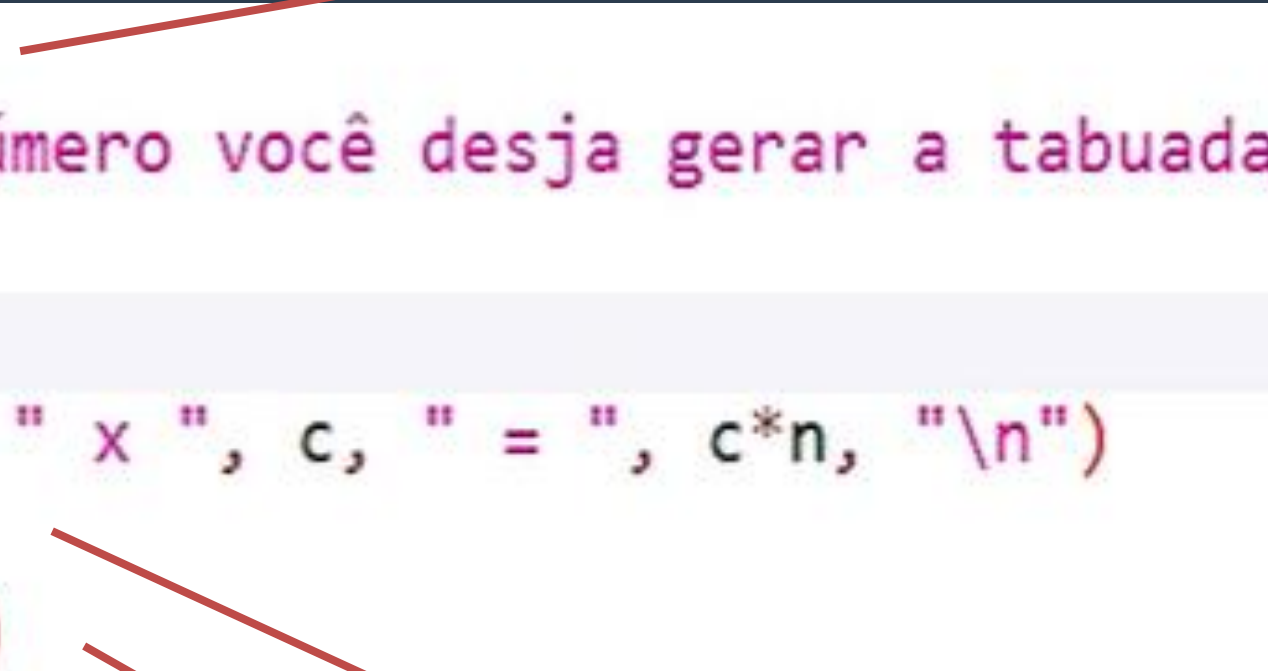
Condição

EXEMPLO - Resolução com Faça Enquanto

Programa que solicita ao usuário para digitar um número n . Logo após, calcula e exibe na tela a tabuada do número. Resolução com estrutura de repetição

Faça enquanto

```
inteiro n, c=0
escreva("Qual número você deseja gerar a tabuada? ")
leia(n)
faca {
    escreva(n, " x ", c, " = ", c*n, "\n")
    c = c + 1
}enquanto(c<=10)
```



Inicialização

Condição

Incremento

```
1  #include <stdio.h>
2  #include <locale.h>
3
4  int main(){
5      setlocale(LC_ALL, "Portuguese");
6
7      int n1=0, c=0;
8      printf("Qual número você deseja gerar a tabuada: ");
9      scanf("%d", &n1);
10
11  do{
12      printf("%d x %d = %d\n", n1, c, n1*c);
13      c++;
14  }while(c <= 10);
15
16  return 0;
17 }
```


Estruturas de Repetição:

PARA

O bloco do comando para é executado enquanto uma condição for verdadeira.

Exemplo:

```
para (variável = valorInicial; variável < valorFinal; variável++)
```

```
{
```

bloco de instruções

```
}
```



Inicialização

Condição

Incremento

EXEMPLO

EXEMPLO DE ESTRUTURA DE REPETIÇÃO PARA

Programa que executa um laço de repetição que imprime "*", até que a condição seja falsa.

```
1 programa
2 {
3     funcao inicio()
4     {
5         inteiro c
6         para(c=1; c<=3; c++)
7         {
8             escreva("Olá Mundo\n")
9         }
10    }
11 }
```

Inicialização

Condição

Incremento

```
1 #include <stdio.h>
2 #include <locale.h>
3
4 int main(){
5     setlocale(LC_ALL, "Portuguese");
6
7     int i;
8     for(i=1; i<=3; i++){
9         printf("Olá Mundo!\n");
10    }
11
12    return 0;
13 }
```

EXEMPLO

EXEMPLO DE ESTRUTURA DE REPETIÇÃO EM C

```
ex1_For.cpp
1  #include <stdio.h>
2  #include <locale.h>
3
4  int main(void)
5  {
6      setlocale(LC_ALL, "Portuguese");
7      int i;
8
9      for(i=0; i<=10; i++)
10     {
11         printf("Valor de i = %d\n", i);
12     }
13
14     return 0;
15 }
```

```
1  #include <stdio.h>
2  #include <locale.h>
3
4  int main(void)
5  {
6      setlocale(LC_ALL, "Portuguese");
7      int i=0;
8
9      while(i<=10)
10     {
11         printf("Valor de i = %d\n", i);
12         i++;
13     }
14
15     return 0;
16 }
```

```
1  #include <stdio.h>
2  #include <locale.h>
3
4  int main(void)
5  {
6      setlocale(LC_ALL, "Portuguese");
7      int i=0;
8
9      do
10     {
11         printf("Valor de i = %d\n", i);
12         i++;
13     }while(i<=10);
14
15     return 0;
16 }
```

Saída na tela

```
Valor de i = 0
Valor de i = 1
Valor de i = 2
Valor de i = 3
Valor de i = 4
Valor de i = 5
Valor de i = 6
Valor de i = 7
Valor de i = 8
Valor de i = 9
Valor de i = 10
```

Estruturas de Repetição:

FOR

</> source code

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int i = 0;
6
7      for(i=0; i<3; i++)
8      {
9          if(i!=2)
10             printf("*\n");
11          else
12             printf("*");
13      }
14
15      return 0;
16 }
```

```
C:\Users\ffass\Desktop\for2.e  ×  +
*
*
*
-----
Process exited after 0.07518
Pressione qualquer tecla para
```

```
1  #include <stdio.h>
2  #include <locale.h>
3
4  int main(){
5      setlocale(LC_ALL, "Portuguese");
6
7      int i=0;
8      for(i=0; i<3; i++){
9          printf("*\n");
10     }
11
12     return 0;
13 }
```

```
C:\Users\ffass\Desktop\for2.e  ×
*
*
*
-----
Process exited after 0.0
```

```
1  #include <stdio.h>
2  #include <locale.h>
3
4  int main(){
5      setlocale(LC_ALL, "Portuguese");
6
7      int i=0;
8      for(i=0; i<3; i++){
9          printf("*");
10     }
11
12     return 0;
13 }
```

```
C:\Users\ffass\Desktop\for2.e  ×
***
-----
Process exited after 0.0
```

Exemplo:

EXEMPLO DE ALGORITMO

Programa em C, que verifica se a senha digitada esta correta e permite o acesso

```
1  #include <stdio.h>
2
3  int main(void) {
4      int senhaDigitada;
5      int senhaFabio = 123;
6
7      printf("Digite sua senha: ");
8      scanf("%d", &senhaDigitada);
9
10     if(senhaDigitada == senhaFabio){
11         printf("Foi permitido o acesso");
12     }
13
14     else
15         printf("Acesso negado");
16
17     return 0;
18 }
```

Exemplo:

EXEMPLO DE ALGORITMO

Programa em C utilizando while, que verifica se a senha digitada esta correta e permite o acesso.

```
1  #include <stdio.h>
2
3  int main(void) {
4      int senhaDigitada;
5      int senhaFabio = 123;
6
7      printf("Digite sua senha: ");
8      scanf("%d", &senhaDigitada);
9
10     while(senhaDigitada != senhaFabio){
11         printf("Acesso negado\n");
12         printf("Digite sua senha novamente: ");
13         scanf("%d", &senhaDigitada);
14     }
15
16     printf("Foi permitido o acesso");
17
18     return 0;
19 }
```

EXERCÍCIO

Faça um programa em "C" que lê dois valores e imprime:

- se o primeiro valor for menor que o segundo, a lista de valores do primeiro até o segundo;
- se o primeiro valor for maior que o segundo, a lista de valores do segundo até o primeiro;
- se ambos forem iguais a mensagem "valores iguais".

Solução

```
#include <stdio.h>
#include <locale.h>
int main() {
    setlocale(LC_ALL, "Portuguese");
    int i=0, n1=0, n2=0;

    printf("Digite o valor 1 = ");
    scanf("%d", &n1);

    printf("Digite o valor 2 = ");
    scanf("%d", &n2);

    if(n1==n2)
        printf("Valores iguais");

    else if(n1 < n2) {
        for(i=n1; i<=n2; i++)
            printf("%d ", i);
    }

    else{
        for(i=n1; i>=n2; i--)
            printf("%d ", i);
    }
    return 0;
}
```


EXERCÍCIO – RESOLUÇÃO

Solução em portugol

```
1 programa
2 {
3
4     funcao inicio()
5     {
6         inteiro i=0, n1=0, n2=0
7
8         escreva("Digite o valor 1 = ")
9         leia(n1)
10
11        escreva("Digite o valor 2 = ")
12        leia(n2)
13
14        se(n1==n2)
15            escreva("Valores iguais")
16
17        senao se(n1<n2){
18            para(i=n1; i<=n2; i++)
19                escreva(i, " ")
20        }
21
22        senao{
23            para(i=n1; i>=n2; i--)
24                escreva(i, " ")
25        }
26    }
27 }
```

EXERCÍCIO

Fazer um programa em "C" que lê o preço de um produto e inflaciona esse preço em 10% se ele for menor que 100 e em 20% se ele for maior ou igual a 100. Imprima o preço final na tela.

EXERCÍCIO – RESOLUÇÃO

```
#include <stdio.h>
#include <locale.h>
#include <stdlib.h>
#include <math.h>
int main()
{
    setlocale(LC_ALL, "Portuguese");
    float preco=0;

    printf("Digite o preço do produto ");
    scanf("%f", &preco);

    if(preco < 100)
    {
        preco = preco*1.1;
        printf("O preço com 10%% = %f", preco);
    }
    else
    {
        preco = preco*1.2;
        printf("O preço com 20%% = %f", preco);
    }

    return 0;
}
```

EXERCÍCIO

Escreva um algoritmo em C que imprima todos os números ímpares do intervalo fechado de 1 a 100.

EXERCÍCIO – RESOLUÇÃO

```
#include <stdio.h>
#include <locale.h>
#include <stdlib.h>
#include <math.h>
int main()
{
    setlocale(LC_ALL, "Portuguese");
    int tamanho = 100;
    int i=0, impar=0;

    for(i = 1; i<=tamanho; i++)
    {
        impar = i%2;

        if(impar != 0)
        {
            printf("%d ", i);
        }
    }

    return 0;
}
```

EXERCÍCIO

Escreva um algoritmo em C que imprima todos os números inteiros de 200 a 100 (em ordem decrescente).

EXERCÍCIO – RESOLUÇÃO

```
#include <stdio.h>
#include <locale.h>
#include <stdlib.h>
#include <math.h>
int main()
{
    setlocale(LC_ALL, "Portuguese");
    int tamanho = 100;
    int i=0, n=0;

    for(i = 200; i>=100; i--)
    {
        printf("%d ", i);
    }

    return 0;
}
```

EXERCÍCIO

EXERCÍCIO DE ESTRUTURA DE REPETIÇÃO ENQUANTO

Faça um programa que solicite ao usuário para digitar um número x. Logo após, calcula e exibe na tela a tabuada do número digitado pelo usuário.

EXERCÍCIO DE ESTRUTURA DE REPETIÇÃO ENQUANTO

Faça um programa que solicite ao usuário para digitar um número x. Logo após, calcula e exhibe na tela a tabuada do número digitado pelo usuário.

sem estrutura de repetição

```
programa
{
    funcao inicio ()
    {
        inteiro x

        escreva ("digite um número: ")
        leia(x)

        escreva ("\n", x, " x 1 = ", x*1)
        escreva ("\n", x, " x 2 = ", x*2)
        escreva ("\n", x, " x 3 = ", x*3)
        escreva ("\n", x, " x 4 = ", x*4)
        escreva ("\n", x, " x 5 = ", x*5)
        escreva ("\n", x, " x 6 = ", x*6)
        escreva ("\n", x, " x 7 = ", x*7)
        escreva ("\n", x, " x 8 = ", x*8)
        escreva ("\n", x, " x 9 = ", x*9)
        escreva ("\n", x, " x 10 = ", x*10)

    }
}
```

EXERCÍCIO – RESOLUÇÃO COM ESTRUTURA DE REPETIÇÃO ENQUANTO

```
programa
{
    funcao inicio ()
    {
        inteiro x, t=1

        escreva ("digite um número: ")
        leia(x)

        enquanto(t<=10)
        {
            escreva ("\n", x, " x ", t, " = ", x*t)
            t=t+1
        }
    }
}
```

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

int main() {
    setlocale(LC_ALL, "Portuguese");
    int x=0, t=0, tabuada=0;

    printf("Digite um número: ");
    scanf("%d", &x);

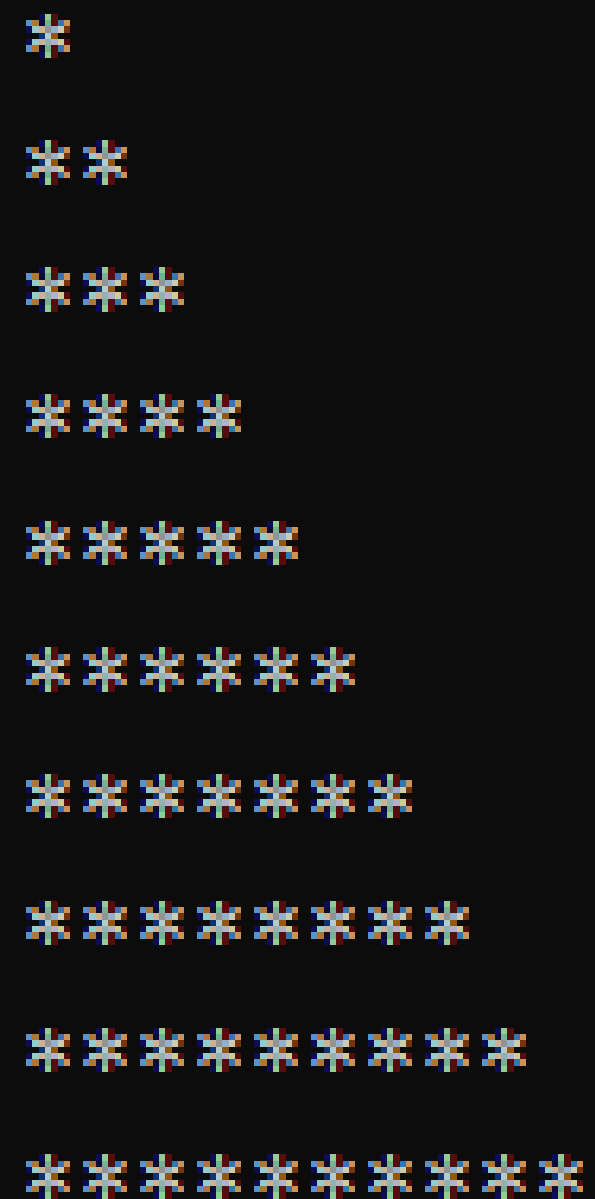
    while (t <= 10)
    {
        tabuada = x*t;
        printf("\n %d x %d = %d", x, t, tabuada);
        t ++;
    }

    return 0;
}
```

EXERCÍCIO

EXERCÍCIO DE ESTRUTURA DE REPETIÇÃO ENQUANTO

Faça um programa que imprime 10 asteriscos (*) em linha e 10 em coluna, formando um triângulo conforme imagem abaixo.



```
*  
* *  
* * *  
* * * *  
* * * * *  
* * * * * *  
* * * * * * *  
* * * * * * * *  
* * * * * * * * *  
* * * * * * * * * *
```

EXERCÍCIO – RESOLUÇÃO

```
1 programa
2 {
3     funcao inicio()
4     {
5         inteiro linha = 1, coluna = 1
6
7         enquanto(linha <= 10)
8         {
9             coluna = 1
10            enquanto(coluna <= linha)
11            {
12                escreva("*")
13                coluna = coluna + 1
14            }
15            escreva("\n")
16            linha = linha + 1
17        }
18    }
19 }
```

```

1  #include <stdio.h>
2  #include <locale.h>
3
4  int main() {
5      setlocale(LC_ALL, "Portuguese");
6      int linha, coluna;
7
8      linha = 1;
9      while(linha <= 10)
10     {
11         coluna = 1;
12
13         while(coluna <= linha)
14         {
15             printf( "*" );
16             coluna = coluna + 1;
17         }
18         printf( "\n" );
19         linha = linha + 1;
20     }
21     return 0;
22 }

```

```

*
*
*
*
*
*
*
*
*
*
*
*
*
*
*
*
*
*
*
*

```

Lista de Exercícios

Fim

A Computação ilumina sua mente

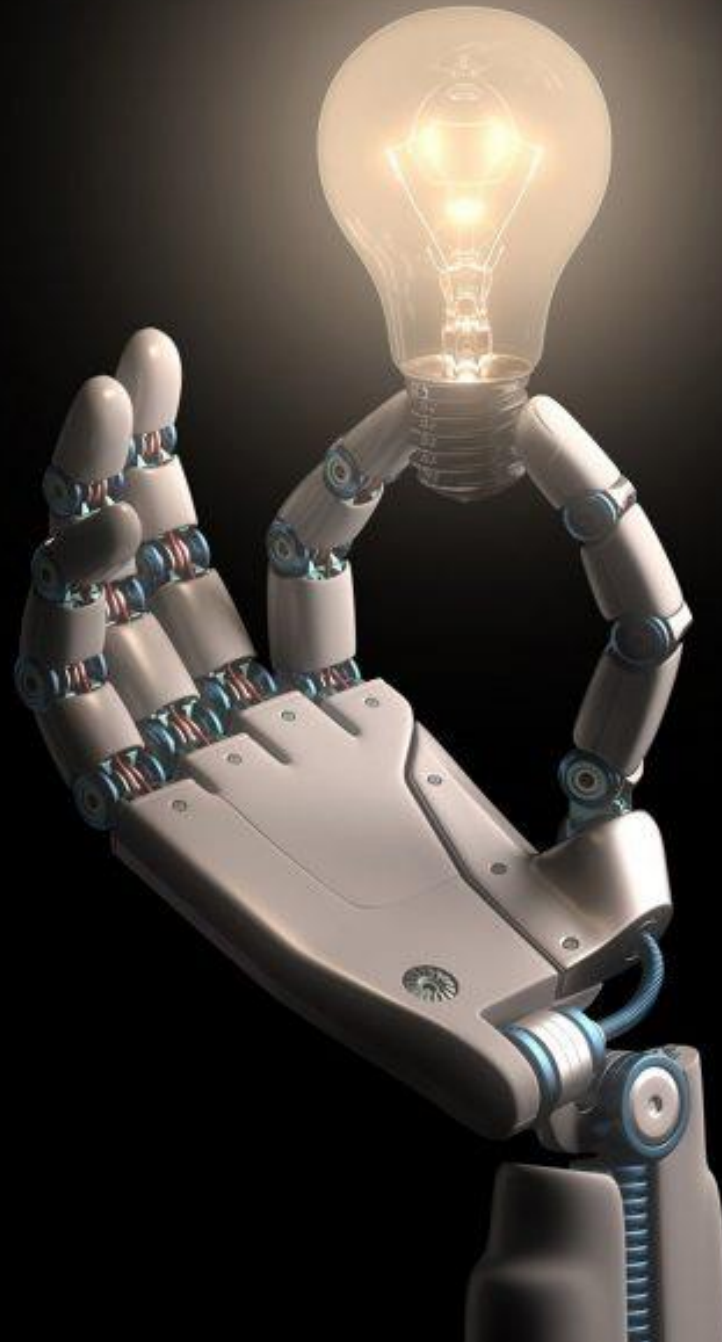


Figura 4 – Computação x robótica (Adaptado)
Fonte: <https://tecnico.ulisboa.pt/pt/tag/robotica/>

Bibliografia

Básica

1. CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. Algoritmos: teoria e prática. 2ª edição. Rio de Janeiro, RJ: Campus, 2002.
2. KNUTH, D. E. The art of computer programming. Upper Saddle River, USA: Addison-Wesley, 2005.
3. SEDGEWICK, R. Algorithms in C: parts 1-4 (fundamental algorithms, data structures, sorting, searching). Reading, USA: Addison-Wesley, 1998.
4. Wes McKinney, Python Para Análise de Dados: Tratamento de Dados com Pandas, NumPy e IPython. Edição Português, Novatec Editora, 9 janeiro 2018.
5. Exemplos da IDE Portugol Studio.

Complementar

Gonick, Larry (1984). Introdução Ilustrada à Computação. São Paulo: Harper & Row do Brasil. p. 115-122. 242 páginas.

Bianchi, Paulo; Bezerra, Milton (1983). Microcomputadores. Arquitetura-Projeto-Programação. Rio de Janeiro: LTC. p. 14-18. 223 páginas. ISBN 85-216-0321-5.

Murdocca, Miles J.; Heuring, Vincent P (2000). Introdução à Arquitetura de Computadores. Rio de Janeiro: Campus. p. 8. 512 páginas. ISBN 85-352-0684-1.

Davis, Martin (2000). «2:Boole Turns Logic into Algebra». Engines of Logic. Mathematicians and the Origin of the Computer (em inglês). New York: W. W. Norton. p. 32. 257 páginas. ISBN 0-393-32229-7.

MONTEIRO, Mário A. Introdução à organização de computadores. São Paulo: LTC, 2012.

NULL, Linda; LOBUR, Julia. Princípios básicos de arquitetura e organização de computadores. São Paulo: Bookman, 2010.

WEBER, Raul Fernando. Fundamentos de arquitetura de computadores. Porto Alegre: Bookman, 2012.

Gary, Michael R., and David S. Johnson: Computers and Intractability: A Guide to the Theory of NP-Completeness. New York: W. H. Freeman & Co., 1979.