



# Algoritmos e Estrutura de Dados

**Profª. Angela Abreu Rosa de Sá, Drª.**

---

*Contato: [angelaabreu@gmail.com](mailto:angelaabreu@gmail.com)*

# Como aprender?



- ◆ **Tem que fazer SENTIDO**
- ◆ **Motivação**
- ◆ **Prazeroso**

# Para que serve?

---



**Manutenção de Título a Pagar**

**Dados Básicos** **Complementar** **Nota Fiscal**

Título a Pagar

Número do Documento <input type="text"/>	<b>Gerar</b>	Data de Vencimento <input type="text"/>	Local Pagamento -> Selecione <-	Parcela <input type="text"/>
Descrição <input type="text"/>		Valor a Pagar <input type="text"/> 0,00	Data Emissão Sex 13/03/2009	Data Protocolo Sex 13/03/2009
Cedente <input type="text"/>		Conta Contábil Crédito Dígito <input type="text"/> <input type="checkbox"/>	Tipo Cobrança -> Selecione <-	
Histórico -> Selecione <-		Conta Contábil Débito Dígito <input type="text"/> <input type="checkbox"/>	Responsável -> Selecione <-	
Status <input type="text"/>			Centro Custo -> Selecione <-	

**Incluir** **Alterar** **Excluir** **Baixar** **Limpar** **Fechar**



# Utilidade







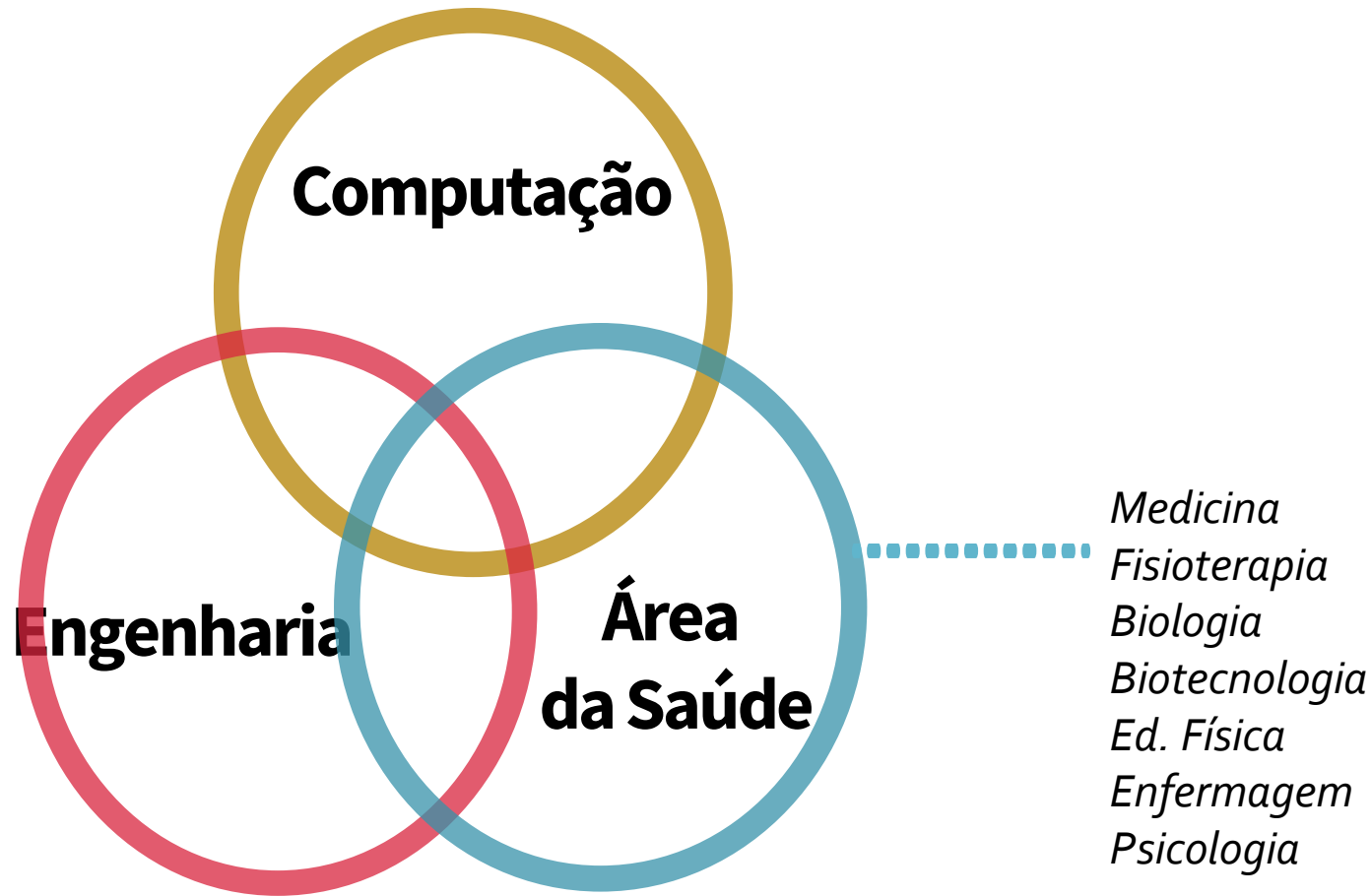
# Algoritmos



```
string sInput;  
int iLength, iN;  
double dblTemp;  
bool again = true;  
  
while (again) {  
    iN = -1;  
    again = false;  
    getline(cin, sInput);  
    system("cls");  
    stringstream(sInput) >> dblTemp;  
    iLength = sInput.length();  
    if (iLength < 4) {  
        again = true;  
        continue;  
    } else if (sInput[iLength - 3] != '.') {  
        again = true;  
        continue;  
    } while (++iN < iLength) {  
        if (isdigit(sInput[iN])) {  
            continue;  
        } else if (iN == (iLength - 3)) {  
            continue;  
        }  
    }  
}
```

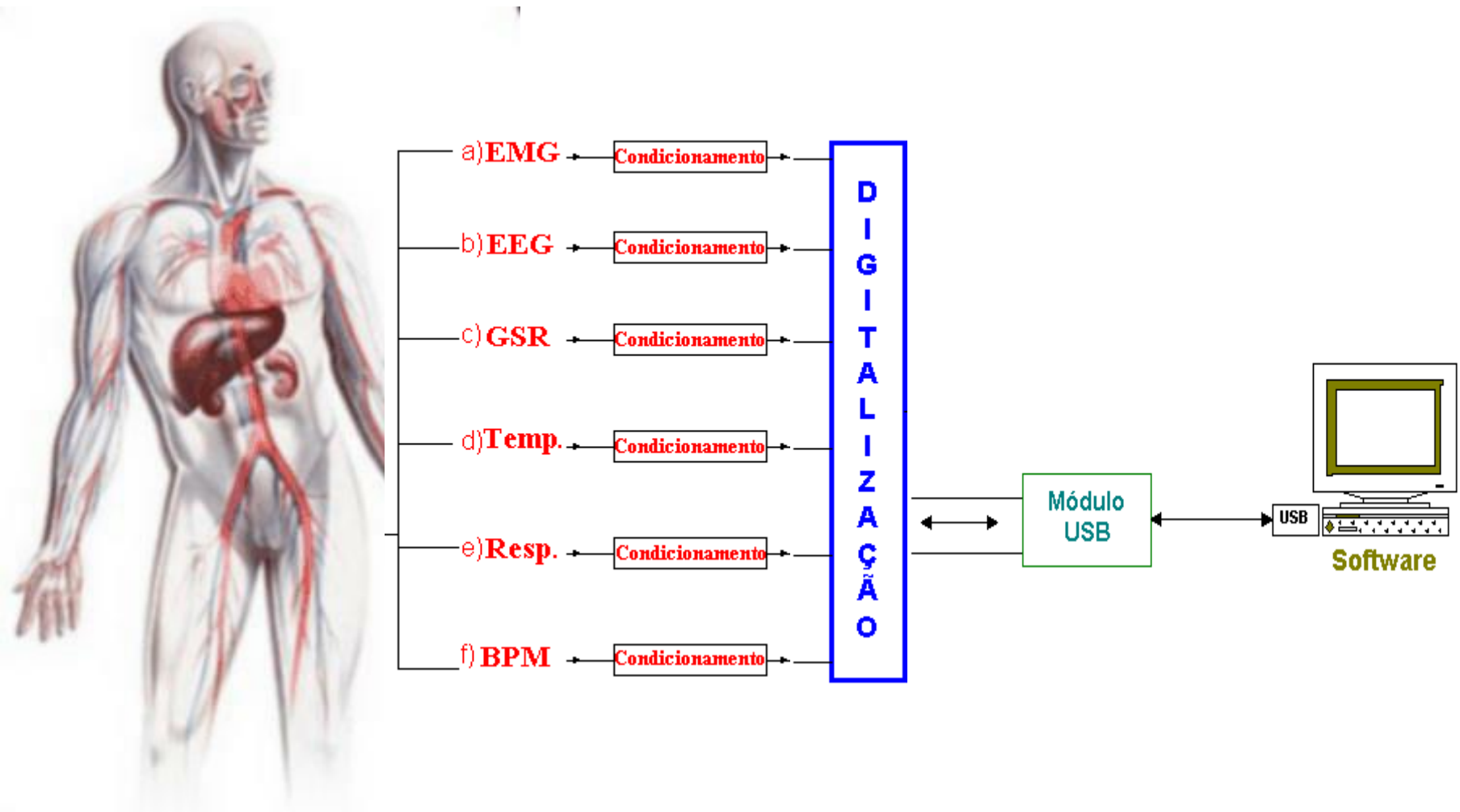
# Engenharia Biomédica

---

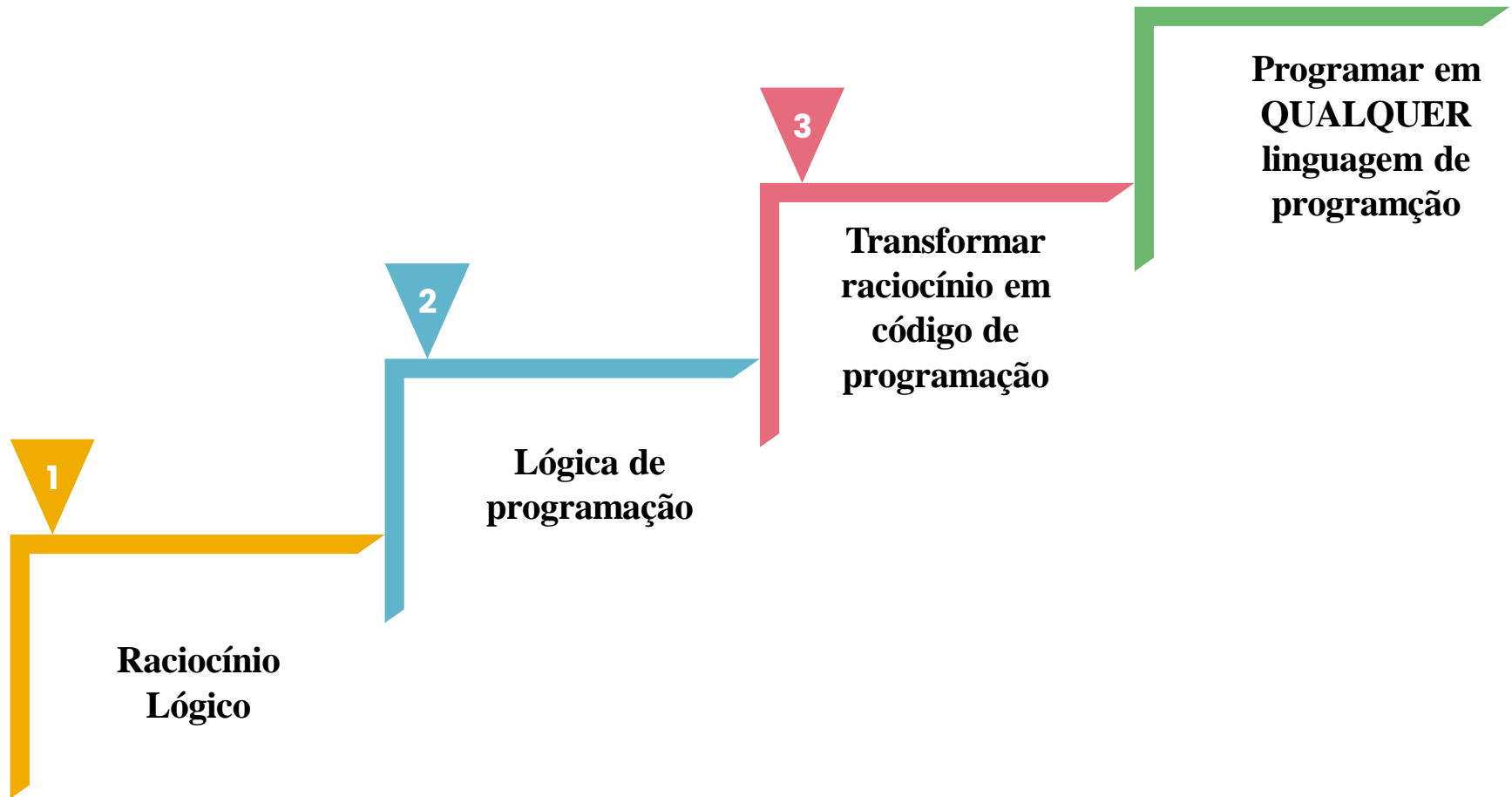




# Atuação Profissional



# O mais importante...



# É fácil?

É um processo **INDIVIDUAL!**



Depende apenas de você!

IMPORTANTE

- Não se compare com o colega;
- Pergunte **QUALQUER** dúvida;
- Pratique sempre...inclusive em casa!
- Não desista!

# Docência





# Ementa



## Horário das Aulas

**Segunda-feira: 19hs - 22hs**

**Intervalo: 20h15 – 20h45**

—→ *Lista de presença*

# Dinâmica

---

**2 Avaliações Oficiais**

1000 pontos

4000 pontos

**Trabalhos: 2500 pontos**

1000 pontos – **Atividade 1**

1500 pontos - **Atividade 2**

**Total: 7500 pontos**

# Dinâmica

AGOSTO / 2022

D	S	T	Q	Q	S	S
	1	2	3	4	5	6
7	8 <sup>C</sup>	9	10	11	12	13
14	15 <sup>M</sup>	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31 <sup>M</sup>			

SETEMBRO / 2022

D	S	T	Q	Q	S	S
				1	2	3
4	5	6	7 <sup>M</sup>	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

Avaliação 1

OUTUBRO / 2022

D	S	T	Q	Q	S	S
						1
2	3	4	5	6	7	8
9	10	11	12 <sup>M</sup>	13	14 <sup>M</sup>	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

NOVEMBRO / 2022

D	S	T	Q	Q	S	S
		1	2 <sup>M</sup>	3	4	5
6	7	8	9	10	11	12
13	14	15 <sup>M</sup>	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

Avaliação 2



# Ementa

## Listas Ligadas

Definição  
Operações  
Listas duplamente ligadas

1

## Pilhas e filas

Definição  
Operações com pilhas  
Operações com filas

2

## Tabelas de espalhamento

Definição  
Operações  
Otimização

3

## Armazenamento associativo

Definição  
Mapas com lista  
Mapas com espalhamento

4

# Algoritmos

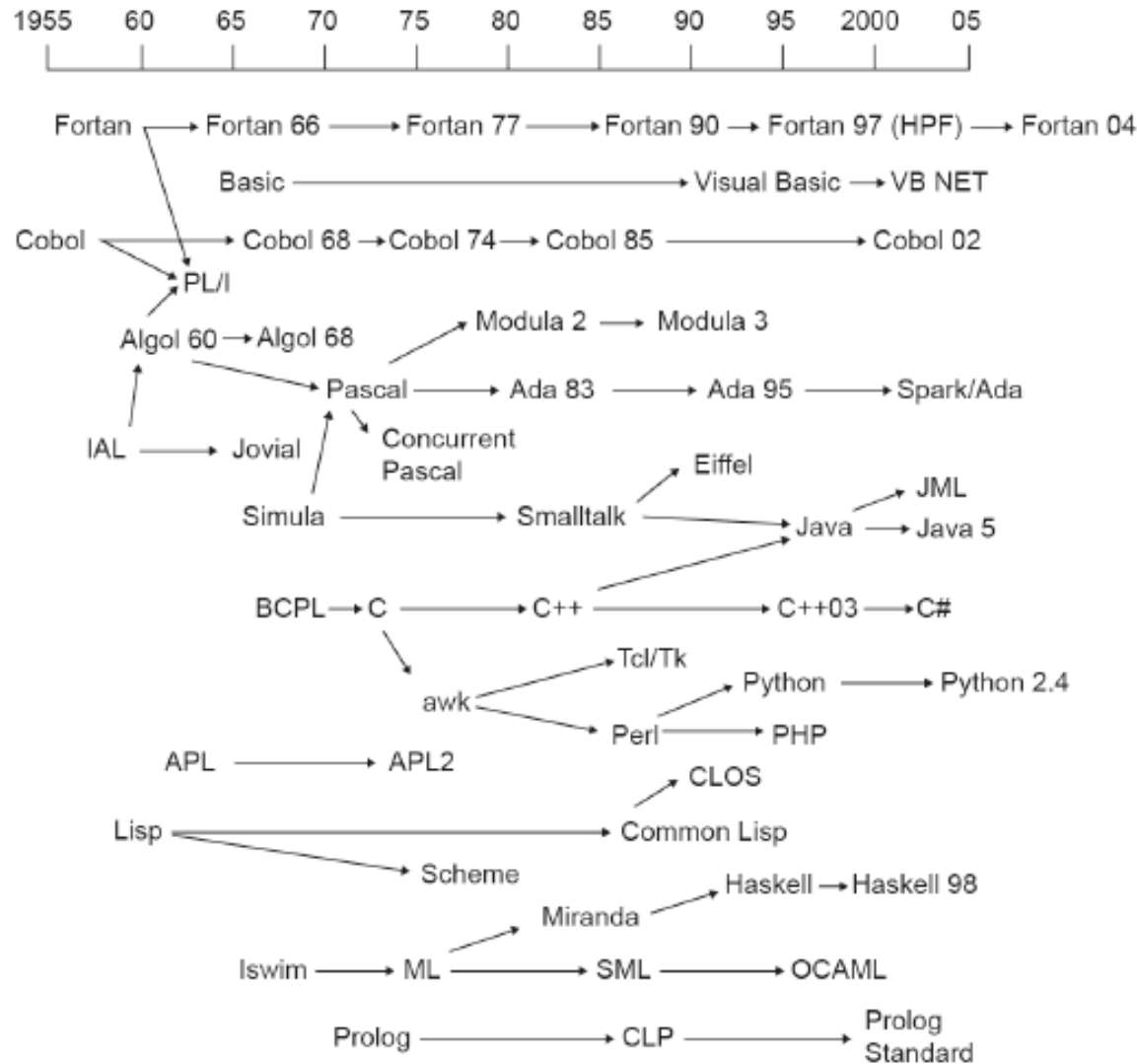
---



**Não** existe algoritmo **único**!

Para um mesmo problema, podemos ter **diversos algoritmos diferentes**!

# Linguagem de Programação



# Linguagem de Programação



Linguagem Alto Nível

Linguagem R  
Linguagem Swift  
Linguagem Google Go  
Linguagem Ruby  
Linguagem PHP  
Linguagem JavaScript  
Linguagem Python  
Linguagem C#  
Linguagem Java  
Linguagem C++  
**Linguagem C**  
Linguagem Pascal  
Linguagem Lisp  
Linguagem Algol  
Linguagem COBOL  
Linguagem Basic  
Linguagem Fortran

Linguagem Assembly

Linguagem de Máquina



# Linguagem de Programação

Most popular in		
Javascript*	16.4 M	Web, backend
Python	11.3 M	DS/ML, IoT apps
Java	9.6 M	Mobile, desktop
C/C++	7.5 M	Embedded, IoT apps
PHP	7.3 M	Web, backend
C#	7.1 M	AR/VR, desktop, games

Fonte: WebSite <https://programadoresbrasil.com.br>, 2021

# Linguagem de Programação



# Linguagem de Programação

---

- Programação Desktop
- Programação Web
- Programação Mobile
- Programação de Jogos

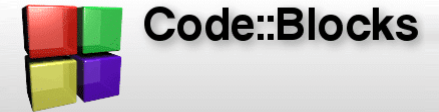
*Depende do contexto....*

# IDE – Ambiente de Desenvolvimento

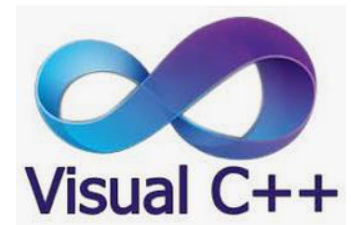
Integrated Development Environment



DevC++



C++ Builder

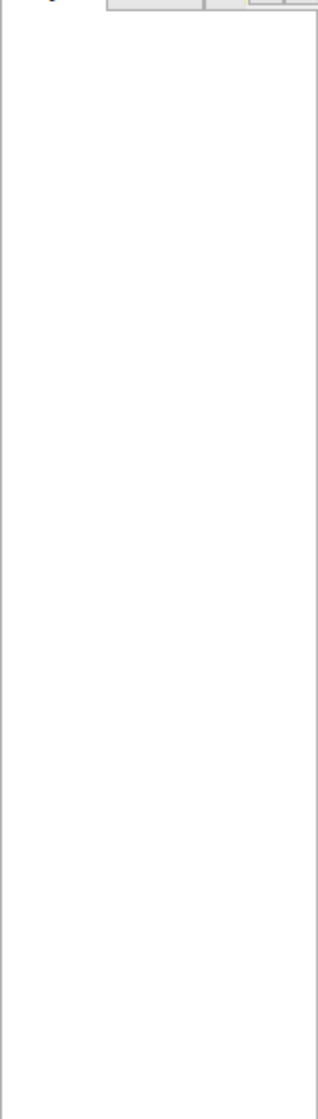




Arquivo Editar Localizar Exibir Projeto Executar Ferramentas AStyle Janela Ajuda



Projeto Classes De ◀ ▶







Arquivo Editar Localizar Exibir Projeto Executar Ferramentas AStyle Janela Ajuda

Novo

Abrir... Ctrl+O

Salvar Ctrl+S

Salvar Como...

Salvar Projeto como...

Salvar Todos Shift+Ctrl+S

Fechar Ctrl+W

Fechar Projeto

Fechar Todas Shift+Ctrl+W

Propriedades

Importar

Exportar

Imprimir Ctrl+P

Configurar Impressão

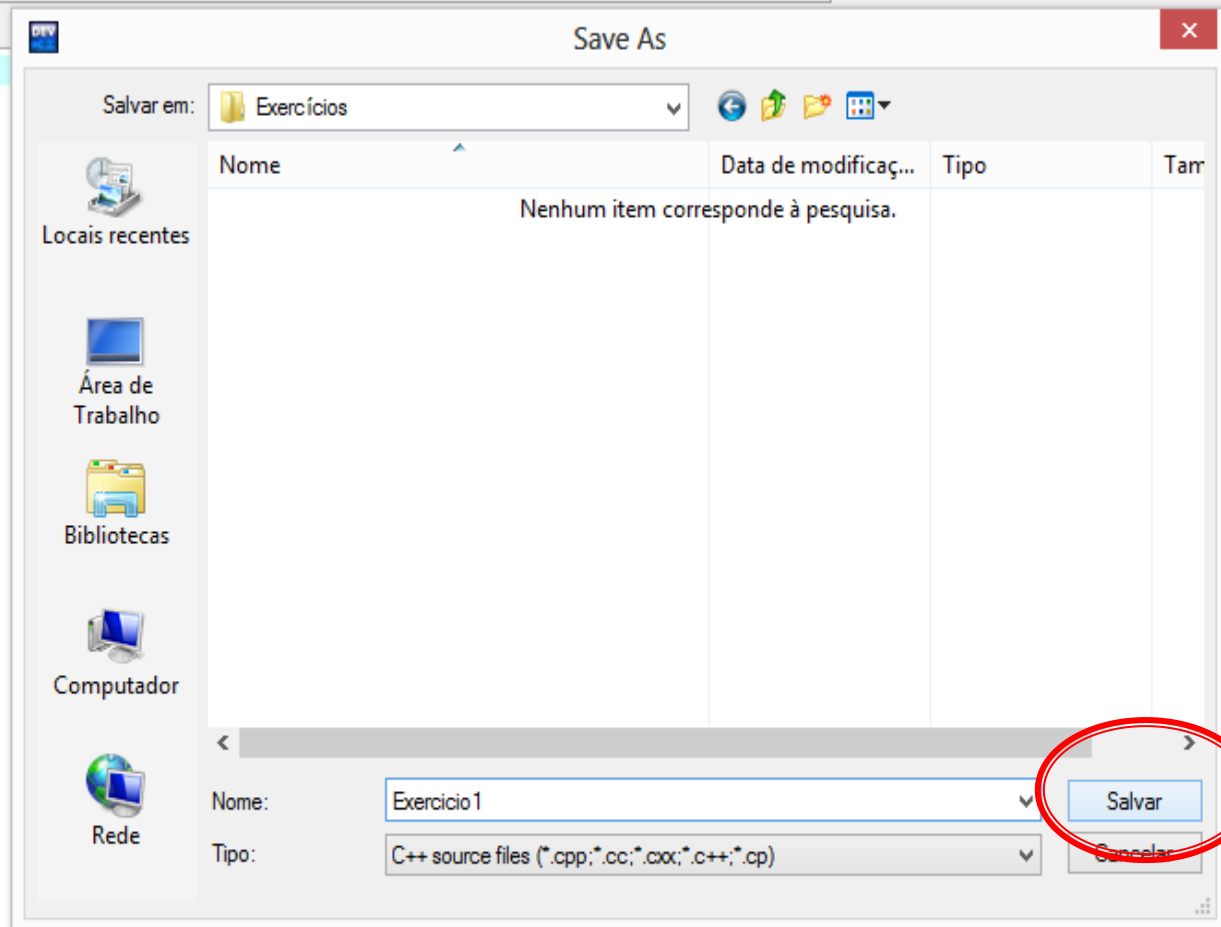
Sair Alt+F4

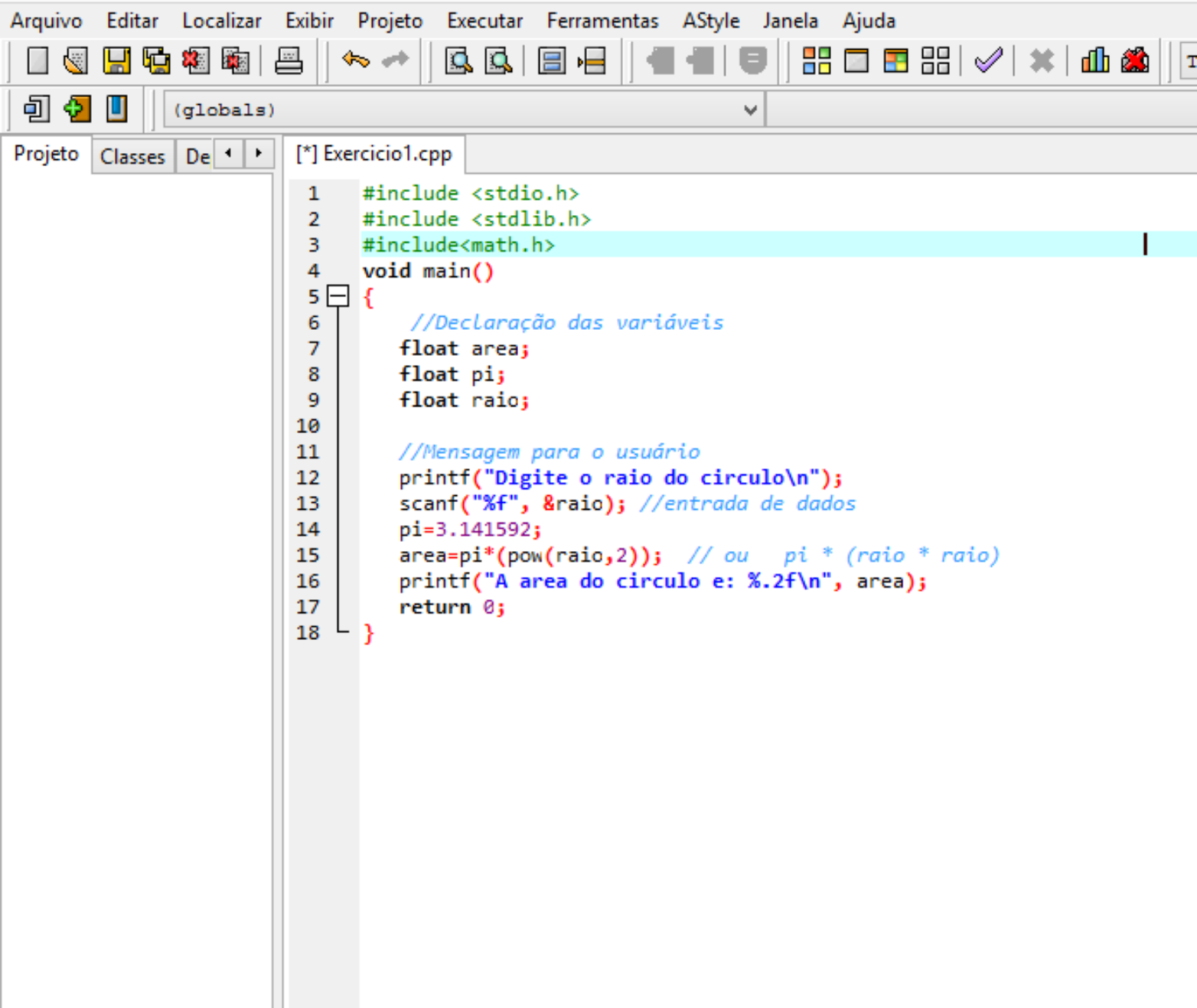
Título1

TDM-GCC 4.9.2 64-bit Release



1





The image shows a screenshot of a C++ IDE. The top menu bar includes 'Arquivo', 'Editar', 'Localizar', 'Exibir', 'Projeto', 'Executar', 'Ferramentas', 'AStyle', 'Janela', and 'Ajuda'. Below the menu is a toolbar with various icons for file operations, editing, and execution. The main editor window displays a C++ program named 'Exercicio1.cpp'. The code calculates the area of a circle by prompting the user for the radius, reading the input, and then using the formula  $area = \pi * r^2$ . The program uses `stdio.h` for input/output, `stdlib.h` for `system`, and `math.h` for the `pow` function. The output is formatted to two decimal places. The line numbers 1 through 18 are visible on the left side of the code editor.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <math.h>
4  void main()
5  {
6      //Declaração das variáveis
7      float area;
8      float pi;
9      float raio;
10
11     //Mensagem para o usuário
12     printf("Digite o raio do circulo\n");
13     scanf("%f", &raio); //entrada de dados
14     pi=3.141592;
15     area=pi*(pow(raio,2)); // ou pi * (raio * raio)
16     printf("A area do circulo e: %.2f\n", area);
17     return 0;
18 }
```

Arquivo Editar Localizar Exibir Projeto Executar Ferramentas AStyle Janela Ajuda

(globals)

Projeto Classes De

[\*] Exercicio1.cpp

```
1 #include <iostream>
2 #include <math>
3 #include <string>
4
5 int main()
6 {
7     // Declaração de variáveis
8     float raio;
9     float area;
10    float pi;
11    // Valor de pi
12    const float PI = 3.14159;
13    pri
14    sca
15    pi =
16    are
17    pri
18    return 0;
19 }
```

Compilar F9  
Executar F10  
Compilar & Executar F11  
Recompilar Tudo F12  
Checar Sintaxe  
Syntax Check Current File Ctrl+F9  
Parâmetros...  
Editar Makefile  
Limpar  
Análise do Perfil  
Apagar Informação de Perfil  
Goto Breakpoint F2  
Criar Breakpoint F4  
Depurar F5  
Parar Execução F6

Compilador Recursos Registro do Compilador Depurador Resultados da Busca Fechar

Abortar Compilação

Shorten compiler paths

Compilation results...

-----

- Errors: 0  
- Warnings: 0  
- Output Filename: C:\Users\angelaabreu\Desktop\Exercícios\Exercicio1.exe  
- Output Size: 151,8984375 KiB  
- Compilation Time: 0,70s

C:\Users\angelaabreu\Desktop\Exercícios\Exercicio1.cpp - Dev

Arquivo Editar Localizar Exibir Projeto Executar Ferramentas AStyle Janela Ajuda

(globals)

Projeto Classes De Exercicio1.cpp

```
1 #inclu
2 #inclu
3 #inclu
4
5 int ma
6 {
7
8 flo
9 flo
10 flo
11
12 //M
13 pri
14 sca
15 pi=
16 are
17 pri
18 retu
19 }
```

Executar F10

Compilar F9

Compilar & Executar F11

Recompilar Tudo F12

✓ Checar Sintaxe

✓ Syntax Check Current File Ctrl+F9

Parâmetros...

Editar Makefile

Limpar

Análise do Perfil

Apagar Informação de Perfil

Goto Breakpoint F2

Criar Breakpoint F4

✓ Depurar F5

Parar Execução F6

TDM-GCC 4.9.2 64-bit Re:

Compilador Recursos Registro do Compilador Depurador Resultados da Busca Fechar

Abortar Compilação

Shorten compiler paths

Compilation results...

-----

- Errors: 0

- Warnings: 0

- Output Filename: C:\Users\angelaabreu\Desktop\Exercícios\Exercicio1.exe

- Output Size: 151,8984375 KiB

- Compilation Time: 0,28s

C:\Users\angelaabreu\Desktop\Exercícios\Exercicio1.exe

```
Digite o raio do circulo
3
A area do circulo e: 28.27

-----
Process exited after 3.465 seconds with return value 0
Pressione qualquer tecla para continuar. . . _
```

Abortar Compilação

Compilation results...

```
-----
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\angelaabreu\Desktop\Exercícios\Exercicio1.exe
- Output Size: 151,8984375 KiB
- Compilation Time: 0,28s
```

Arquivo Editar Localizar Exibir Projeto Executar Ferramentas AStyle Janela Ajuda

(globals)

Projeto Classes De Exercicio1.cpp

```

1  #inclu
2  #inclu
3  #inclu
4
5  int ma
6  {
7
8  flo
9  flo
10 flo
11
12 //M
13 pri
14 sca
15 pi=
16 are
17 pri
18 return
19 }

```

Executar

- Compilar F9
- Executar F10
- Compilar & Executar F11
- Recompilar Tudo F12
- ✓ Checar Sintaxe
- ✓ Syntax Check Current File Ctrl+F9
- Parâmetros...
- Editar Makefile
- Limpar
- Análise do Perfil
- Apagar Informação de Perfil
- Goto Breakpoint F2
- Criar Breakpoint F4
- ✓ Depurar F5
- Parar Execução F6

TDM-GCC 4.9.2 64-bit Release

Compilador Recursos Registro do Compilador Depurador Resultados da Busca Fechar

Abortar Compilação

Compilation results...

```

-----
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\angelaabreu\Desktop\Exercícios\Exercicio1.exe
- Output Size: 151,8984375 KiB
- Compilation Time: 0,28s

```

☐ Shorten compiler paths

# Variáveis

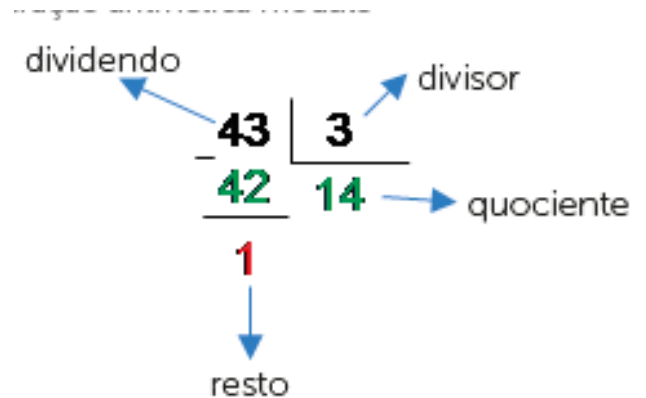
Tabela 2.2 | Tipos de variáveis e sua capacidade

Tipo	Tamanho (byte)	Valores
<i>int</i>	4	-2.147.423.648 até 2.147.423.648
<i>float</i>	4	$-3,4^{38}$ até $3,4^{38}$
<i>double</i>	8	$-1,7^{308}$ até $1,7^{308}$
<i>char</i>	1	-128 até 127
<i>unsigned int</i>	4	0 até 4.294.967.295
<i>short int</i>	2	-32.768 até 32.767
<i>long double</i>	16	$-3,4^{4932}$ até $1,1^{4932}$



# Operadores Matemáticos

Operador	Descrição	Exemplo	Resultado
+	Soma	$4 + 2$	6
-	Subtração	$4 - 2$	2
*	Multiplicação	$4 * 2$	8
/	Divisão	$4 / 2$	2
=	Atribuição	$x = 4$	$x = 4$
%	Módulo	$4 \% 2$	0



- 1° parênteses;
- 2° potenciação e radiciação;
- 3° multiplicação, divisão e módulo;
- 4° soma e subtração;

Ordem de execução

# Operadores Matemáticos

Operador	Descrição	Exemplo	Resultado
++	Pós-incremento	$x++$	$x + 1$
++	Pré-incremento	$++x$	$x + 1$
--	Pós-decremento	$y--$	$y - 1$
--	Pré-decremento	$--y$	$y - 1$

# Operadores Relacionais

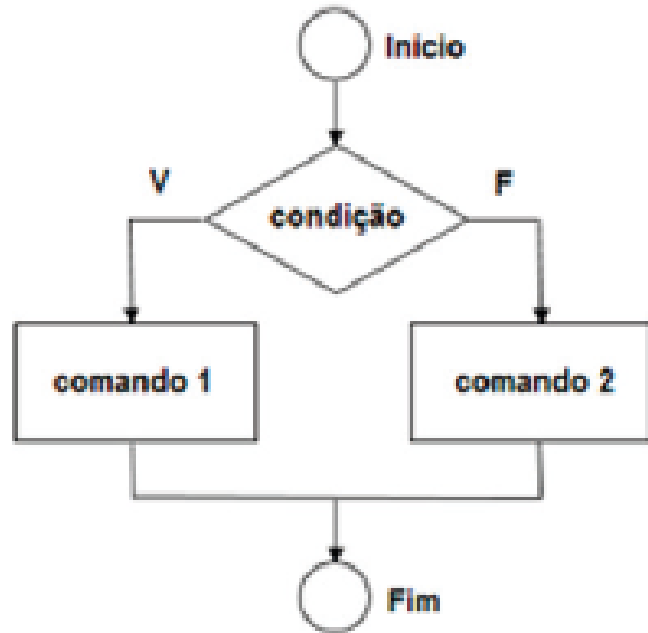
Operador em Linguagem C	Descrição
>	Maior
<	Menor
>=	Maior ou igual
<=	Menor ou igual
==	Igual
!=	Diferente

# Operadores Lógicos

---

Operador em Linguagem C	Operador em algoritmos
&&	Lógico E - conjunção
	Lógico OU - disjunção
!	Lógico NÃO - negação

# if...else



```
if <(condição)>
{
<primeiro conjunto de comandos>;
}
else
{
<segundo conjunto de comandos>;
}
```

# if...else

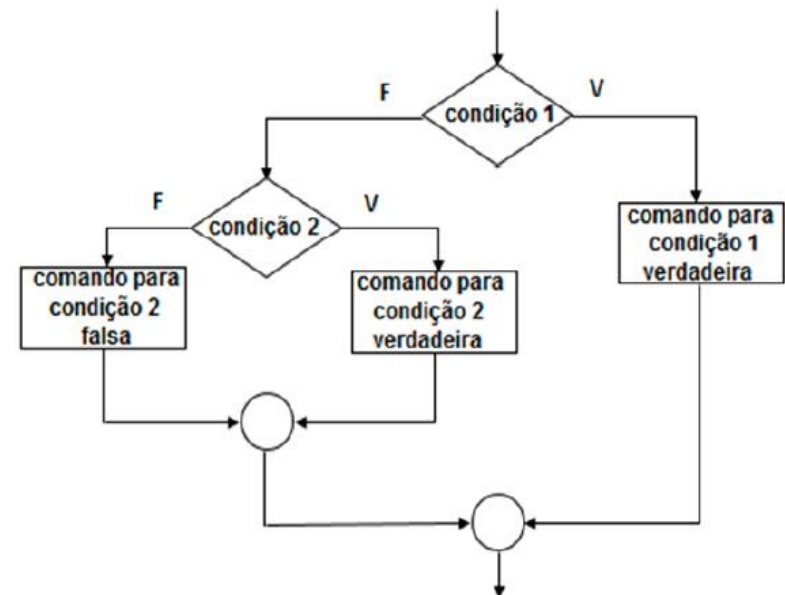
```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main()
{
    float idade;
    printf("Digite sua idade: \n");
    scanf("%f", &idade);
    if (idade >= 18)
    {
        printf("Voce ja pode tirar sua carteira de 9. Habilitacao, voce e maior de 18");
    }
    else {
        printf("Você ainda tem que andar no banco do carona!");
    }

    return 0 ;
}
```

# if...else

```
if(x > 100)
{
    printf ("Maior do que 100");
} else if(x <= 100)
{
    if (x >= 50)
    {
        printf ("Maior do que 50");
    }
    else if(x > 20)
    {
        printf ("Maior do que 20 e menor do que 50");
    }
    else printf ("Menor ou igual a 20");
}
```



# if...else Switch Case

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

int main()
{
    char x;
    float valor, desconto, total;

    printf("Digite a letra que representa o seu deconto de acordo com a cor: \n");
    printf("a - azul \n");
    printf("b - branco \n");
    printf("v - vermelho \n");
    scanf("%s", &x);
    printf("Digite o valor da compra:");
    scanf("%f", &valor);

    if(x == 'a')
    {
        printf("Você escolheu AZUL\n");
        desconto = valor * 0.30;
        total = valor - desconto;
        printf("O valor da sua compra é %.2f \n", total);
    }
    else if(x == 'b')
    {
        printf("Você escolheu BRANCO\n");
        desconto = valor * 0.20;
        total = valor - desconto;
        printf("O valor da sua compra é %.2f \n", total);
    }
    else if(x == 'v')
    {
        printf("Você escolheu VERMELHO\n");
        desconto = valor * 0.10;
        total = valor - desconto;
        printf("O valor da sua compra é %.2f \n", total);
    }
    else printf("**** OPÇÃO INVÁLIDA***** ");

    return 1 ;
}
```



# if...else

# Switch Case

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

int main()
{
    char x;
    float valor, desconto, total;

    printf("Digite a letra que representa o seu deconto de acordo com a cor: \n");
    printf("a - azul \n");
    printf("b - branco \n");
    printf("v - vermelho \n");
    scanf("%s", &x);
    printf("Digite o valor da compra:");
    scanf("%f", &valor);
```

```
    switch(x)
    {
        case 'a': printf("Você escolheu AZUL\n");
                  desconto = valor * 0.30;
                  total = valor - desconto;
                  printf("O valor da sua compra é %.2f \n", total);
                  break;

        case 'b': printf("Você escolheu BRANCO\n");
                  desconto = valor * 0.20;
                  total = valor - desconto;
                  printf("O valor da sua compra é %.2f \n", total);
                  break;

        case 'v': printf("Você escolheu VERMELHO\n");
                  desconto = valor * 0.10;
                  total = valor - desconto;
                  printf("O valor da sua compra é %.2f \n", total);
                  break;

        default: printf("*** OPÇÃO INVÁLIDA***** ");
    }
}
```

```
return 1 ;
```

```
}
```

# if...else

```
int main()

{

    int lado1,lado2,lado3;

    printf("***** CLASSIFICAÇÃO DO TRIÂNGULO ***** \n\n\n");
    printf("Digite o lado 1:");
    scanf("%d",&lado1);
    printf("Digite o lado 2:");
    scanf("%d",&lado2);
    printf("Digite o lado 3:");
    scanf("%d",&lado3);

    //verificar se as medidas formam um trinagulo

    if( (lado1 < (lado2+lado3)) && (lado2 < (lado1+ lado3)) && (lado3 < (lado1+ lado2)) )
    {
        if( (lado1 == lado2) && (lado1==lado3) && (lado2 == lado3))
        printf ("O triângulo é EQUILÁTERO!");

        else if ( (lado1 == lado2) || (lado1 == lado3) || (lado2 == lado3) )
        printf ("O triângulo é ISÓSCELES!");

        else      printf ("O triângulo é ESCALENO!");

    } else printf("As medidas fornecidas %d, %d, %d não formam um triângulo",lado1,lado2,lado3);

    return 1 ;
}
```

# if...else

Tabela 3.1 | Descontos INSS

SALÁRIO DE CONTRIBUIÇÃO (R\$)	ALÍQUOTA / INSS
até 1.693,72	8%
de 1.693,73 até 2.822,90	9%
de 2.822,91 até 5.646,80	11%
Acima de 5.646,80	R\$ 621.04 (invariavelmente)

Fonte: elaborada pelo autor.

**Calcular o salário líquido do funcionário considerando as seguintes Alíquotas do INSS e do IR**

Tabela 3.2 | Descontos IR

SALÁRIO (R\$)	ALÍQUOTA / IR
Até 1.903,98	–
De 1.903,99 até 2.826,65	7,5%
De 2.826,66 até 3.751,05	15,0%
De 3.751,06 até 4.664,68	22,5%
Acima de 4.664,68	27,5%

Fonte: elaborada pelo autor.

# if...else

```
int main()
{
    float salariobruto, INSS, IR, salarioliquido;

    printf("***** CALCULO DE SALÁRIO LIGUIDO***** \n\n\n");
    printf("Digite o valor do salário bruto (REAIS):");
    scanf("%f",&salariobruto);

    //Calcular o valor do desconto do INSS
    if( salariobruto <= 1693.72)
    {
        INSS = salariobruto * 0.08;
    }
    else if ( (salariobruto >= 1693.73) && ( salariobruto <= 2822.90) )
    {
        INSS = salariobruto * 0.09;
    }
    else if( (salariobruto >= 2822.91) && ( salariobruto <= 5646.80) )
    {
        INSS = salariobruto * 0.11;
    }
    else INSS = 621.04;
}
```

SALÁRIO DE CONTRIBUIÇÃO (R\$)	ALÍQUOTA / INSS
até 1.693,72	8%
de 1.693,73 até 2.822,90	9%
de 2.822,91 até 5.646,80	11%
Acima de 5.646,80	R\$ 621.04 (invariavelmente)

# if...else

```
//calcular o valor do desconto do IR
if(salariobruto <= 1903.98)
{
    IR = salariobruto * 0;
}
else if ( ( salariobruto >= 1903.99 ) && (salariobruto <= 2826.65) )
{
    IR = salariobruto * 0.075;
}
else if ( ( salariobruto >= 2826.66 ) && (salariobruto <= 3751.05) )
{
    IR = salariobruto * 0.15;
}
else if ( ( salariobruto >= 3751.06 ) && (salariobruto <= 4664.38) )
{
    IR = salariobruto * 0.225;
}
else if (salariobruto > 4664.69)
{
    IR = salariobruto * 0.275;
}
```

//calculo do salário líquido

```
salarioliquido = salariobruto - INSS - IR;

printf("O desconto do INSS é %f:\n", INSS);
printf("O desconto do IR é %f: \n", IR);
printf("O salário líquido é %f:", salarioliquido);

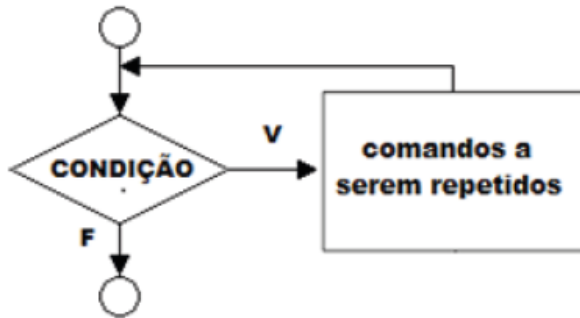
return 1 ;
```

Tabela 3.2 | Descontos IR

SALÁRIO (R\$)	ALÍQUOTA / IR
Até 1.903,98	–
De 1.903,99 até 2.826,65	7,5%
De 2.826,66 até 3.751,05	15,0%
De 3.751,06 até 4.664,68	22,5%
Acima de 4.664,68	27,5%

# Estrutura de Repetição Condicional

Figura 3.4 | Fluxograma do comando *while*

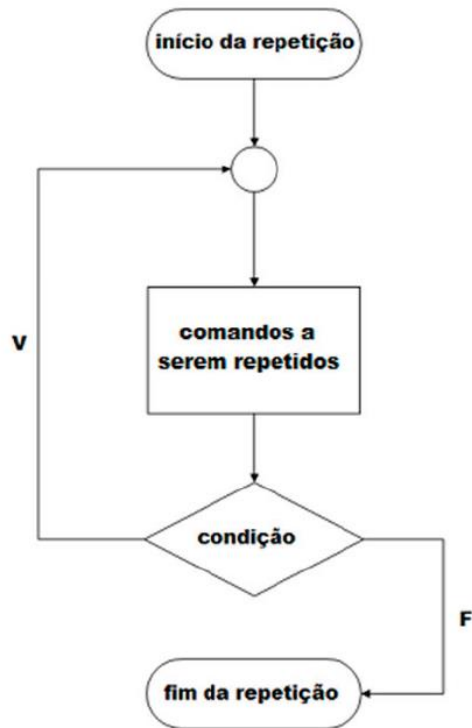


```
while  
(<condição>  
{  
Comando 1;  
Comando 2;  
Comando n;  
}
```

**Enquanto**

**While**

# Do... While



```
Do  
  
{  
  
comandos;  
  
}  
  
while (condição);
```

# Do ... While

```
float soma=0;
float valor;
int opcao;
do {
    printf("\n Digite uma Operacao");
    printf("\n 1. Deposito");
    printf("\n 2. Saque");
    printf("\n 3. Saldo");
    printf("\n 4. Sair");
    printf("\n Opcao? ");
    scanf("%d", &opcao);

    switch(opcao)
    {
        case 1: printf("\n Valor do deposito? ");
                scanf("%f", &valor);
                soma=soma+valor;
                break;
        case 2: printf("\n Valor do saque? ");
                scanf("%f", &valor);
                soma=soma-valor;
                break;
        case 3: printf("\n Saldo atual = R$ %.2f \n", soma);
                break;
        default: if(opcao!=4)
                  printf("\n Opcao Invalida! \n");
    }
}
while (opcao!=4);

printf("Fim das operacoes. \n\n");
```

# While

```
float soma=0;
float valor;
int opcao;
opcao = 1;
while (opcao != 4)
{
    printf("\n Digite uma Operacao");
    printf("\n 1. Deposito");
    printf("\n 2. Saque");
    printf("\n 3. Saldo");
    printf("\n 4. Sair");
    printf("\n Opcao? ");
    scanf("%d", &opcao);

    switch(opcao)
    {
        case 1: printf("\n Valor do deposito? ");
                scanf("%f", &valor);
                soma=soma+valor;
                break;
        case 2: printf("\n Valor do saque? ");
                scanf("%f", &valor);
                soma=soma-valor;
                break;
        case 3: printf("\n Saldo atual = R$ %.2f \n", soma);
                break;
        default: if(opcao!=4)
                  printf("\n Opcao Invalida! \n");
    }
}

printf("Fim das operacoes. \n\n");
```



# Vetores

```
<tipo> <nome_do_vetor>[tamanho];
```

```
int idade;
```

```
int idade[10];
```

35	69	10	25	37	52	36	15	41	23
----	----	----	----	----	----	----	----	----	----

Índice/posição

0 1 2 3 4 5 6 7 8 9

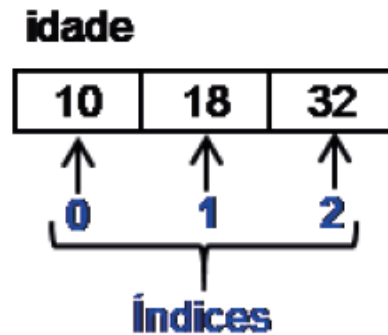
```
idade [0] = 35;
```

```
Idade [8] = 41;
```

```
int x;
```

```
x = idade[2] + idade[9];
```

# Vetores



Valor de idade → Depende do índice

idade[0] = 10

idade[1] = 18

idade[2] = 32

```
printf("Digite uma idade: ");  
scanf("%d", &idade[1]);
```

# Matriz

Dia	Temperatura (°C )
1	26,1
2	27,7
3	30,0
4	32,3
5	27,6
6	29,5
7	29,9

```
<tipo> <nome_da_matriz>[linhas][colunas];
```

```
float matriz_temperatura[7][2]
```

```
matriz_temperatura[0][0] = 1;  
matriz_temperatura[0][1] = 26.1;
```

```
matriz_temperatura[6][0] = 7;  
matriz_temperatura[6][1] = 29.9;
```

# Matriz

		Nota 1	Nota 2
		coluna 0	coluna 1
Aluno 1	linha 0 →	10,0	8,5
Aluno 2	linha 1 →	5,5	2,7
Aluno 3	linha 2 →	4,0	10,0

```
#include<stdio.h>
main() {
    float notas[3][2];

    //aluno 1
    notas[0][0] = 10;
    notas[0][1] = 8.5;

    //aluno 2
    notas[1][0] = 5.5;
    notas[1][1] = 2.7;

    //aluno 3
    notas[2][0] = 4;
    notas[2][1] = 10;
}
```

```
printf("Digite uma nota: ");
scanf("%f",&nota[1][0]);
printf("Nota digitada: %.2f",nota[1][0]);
```

# Structs

```
struct <nome>{  
    <tipo> <nome_da_variavel1>;  
    <tipo> <nome_da_variavel2>;  
    ...  
};
```

```
1.  #include<stdio.h>  
2.  struct automovel{  
3.      char modelo[20];  
4.      int ano;  
5.      float valor;  
6.  };  
7.  main(){  
8.      struct automovel dadosAutomovel1;  
9.      printf("\n Digite o modelo do automovel: ");  
10.     scanf("%s",dadosAutomovel1.modelo);  
11.     printf("\n Digite o ano do automovel: ");  
12.     scanf("%d",&dadosAutomovel1.ano);  
13.     printf("\n Digite o valor do automovel: ");  
14.     scanf("%f",&dadosAutomovel1.valor);  
  
15.     printf("\n Dados atribuidos");  
16.     printf("\n %s",dadosAutomovel1.modelo);  
17.     printf("\n %d",dadosAutomovel1.ano);  
18.     printf("\n %f",dadosAutomovel1.valor);  
19. }
```

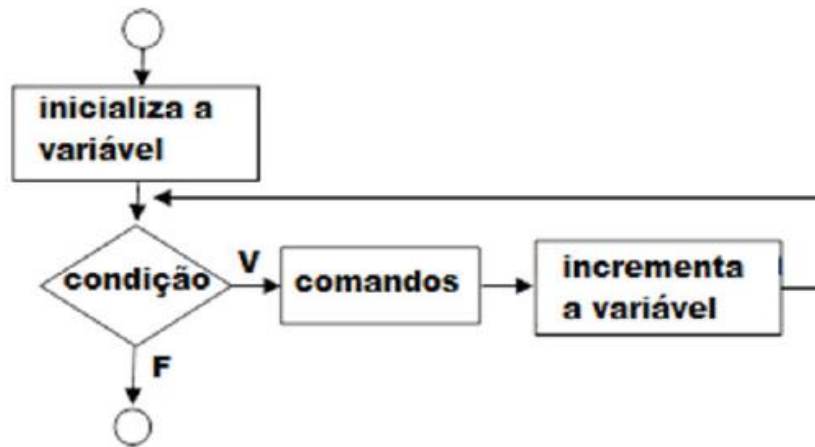
struct automovel dadosautomovel [10];

Dadosautomovel[0].modelo = “ Jeep”;

Dadosautomovel[0].ano = 2021;

Dadosautomovel[0].valor = 180000;

# Estrutura de repetição Determinística: for



```
for( inicialização; condição de parada; incremento)
{
}

```

# Estrutura de repetição Determinística: for

```
#include <stdio.h>
int main(void)
{
    int vetornumero[5];

    for (int i = 0; i < 5; i++)
    {
        printf("\nDigite um numero da posicao %d :", i);
        scanf("%d", &vetornumero[i]);
    }

    printf("\n Os números digitados são: ");
    for (int i = 0; i < 5; i++)
    {
        printf("\n %d", vetornumero[i]);
    }

    return(0);
}
```

```
Digite um numero da posicao 0 :6
Digite um numero da posicao 1 :5
Digite um numero da posicao 2 :6
Digite um numero da posicao 3 :5
Digite um numero da posicao 4 :6

Os n-meros digitados são:
6
5
6
5
6
-----
```

```

#include <stdio.h>
int main(void)
{

int linha,coluna;
int matriz[3][3];
for (linha=0; linha<3; linha++)
{
    for (coluna=0; coluna<3;coluna++)
    {
        printf("Digitar os valores da matriz para: linha %,  coluna %d: ",linha,coluna);
        scanf("%d", &matriz[linha][coluna]);
    }
}

printf("Veja a sua Matriz\n");
for (linha=0;linha<=2;linha++)
{
    for (coluna=0;coluna<3;coluna++)
        printf("%d\t",matriz[linha][coluna]);

    printf("\n");
}
}

```

```

Digitar os valores da matriz para: linha 0,  coluna 0: 2
Digitar os valores da matriz para: linha 0,  coluna 1: 2
Digitar os valores da matriz para: linha 0,  coluna 2: 3
Digitar os valores da matriz para: linha 1,  coluna 0: 5
Digitar os valores da matriz para: linha 1,  coluna 1: 5
Digitar os valores da matriz para: linha 1,  coluna 2: 5
Digitar os valores da matriz para: linha 2,  coluna 0: 6
Digitar os valores da matriz para: linha 2,  coluna 1: 8
Digitar os valores da matriz para: linha 2,  coluna 2: 9
Veja a sua Matriz
2      2      3
5      5      5
6      8      9

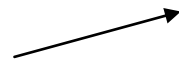
```



# Procedimentos e Funções

```
<tipo de retorno> <nome> (<parâmetros>)  
{  
    <Comandos da função>  
    <Retorno> (não obrigatório)  
}
```

procedimento



**<tipo de retorno>** – Obrigatório. → void | int | float | double | char | \*

**<nome>** – Obrigatório. Obs: mesmas regras para nomes de variáveis

**<parênteses depois do nome>** – Obrigatório.

**<parâmetros>** – Opcional.

**<comandos da função>** – Obrigatório.

**<retorno>** – Quando o tipo de retorno for *void* esse parâmetro não precisa ser usado, porém, quando não for *void* é obrigatório

# Funções

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

```
int somar()  
{  
    return 2 + 3;  
}
```

```
int main()  
{  
    int resultado = 0;  
    resultado = somar();  
    printf("O resultado da funcao e = %d",resultado);  
    return 0;  
}
```

```
O resultado da funcao e = 5
```

# Procedimento

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

```
void somar()  
{  
    int x, y, resultado;  
    x = 3;  
    y = 2;  
    resultado = x + y;  
  
    printf("O resultado da funcao e = %d", resultado);  
  
    return;  
}  
  
int main()  
{  
  
    somar();  
  
    return 0;  
}
```

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

```
float calcular()
```

```
{
```

```
    float num;
```

```
    printf("Digite um numero: ");
```

```
    scanf("%f",&num);
```

```
    return num*num;
```

```
}
```

```
int main()
```

```
{
```

```
    float resultado = 0;
```

```
    resultado = calcular();
```

```
    printf("A potencia do numero digitado = %.2f ",resultado);
```

```
    return 0;
```

```
}
```

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

```
int main()
```

```
{
```

```
    float resultado = 0;
```

```
    float num;
```

```
    printf("Digite um numero: ");
```

```
    scanf("%f",&num);
```

```
    resultado = num*num;
```

```
    printf("A potencia do numero digitado = %.2f ",resultado);
```

```
    return 0;
```

```
}
```

```
Digite um numero: 5
A potencia do numero digitado = 25.00
-----
```

# Passagem de parâmetros para Funções

```
<tipo de retorno> <nome> (<parâmetros>)  
{  
    <Comandos da função>  
    <Retorno> (não obrigatório)  
}
```

```
int somar()  
{ int x, y, resultado;  
  x = 2;  
  y = 3;  
  resultado = x + y;  
  return x + y;  
}
```

```
int somar(int x, int y)  
{  
  int resultado;  
  resultado = x + y;  
  return x + y;  
}
```

# Passagem de parâmetros por valor

```
#include<stdio.h>

int testar(int n1, int n2)
{
    n1 = -1;
    n2 = -2;
    printf("\n\n Valores dentro da funcao testar(): ");
    printf("\n n1 = %d e n2 = %d",n1,n2);
    return 0;
}

int main()
{
    int n1 = 10;
    int n2 = 20;
    printf("\n\n Valores antes de chamar a funcao:");
    printf("\n n1 = %d e n2 = %d",n1,n2);
    testar(n1,n2);
    printf("\n\n Valores depois de chamar a funcao: ");
    printf("\n n1 = %d e n2 = %d",n1,n2);
    return 0;
}
```

Valores antes de chamar a funcao:  
n1 = 10 e n2 = 20

Valores dentro da funcao testar():  
n1 = -1 e n2 = -2

Valores depois de chamar a funcao:  
n1 = 10 e n2 = 20

# Passagem de parâmetros por referência

```
#include<stdio.h>

int testar(int* n1, int* n2)
{
    *n1 = -1;
    *n2 = -2;
    printf("\n\n Valores dentro da funcao testar(): ");
    printf("\n n1 = %d e n2 = %d",*n1,*n2);
    return 0;
}

int main()
{
    int n1 = 10;
    int n2 = 20;
    printf("\n\n Valores antes de chamar a funcao:");
    printf("\n n1 = %d e n2 = %d",n1,n2);
    testar(&n1,&n2);
    printf("\n\n Valores depois de chamar a funcao: ");
    printf("\n n1 = %d e n2 = %d",n1,n2);
    return 0;
}
```

```
Valores antes de chamar a funcao:
n1 = 10 e n2 = 20

Valores dentro da funcao testar():
n1 = -1 e n2 = -2

Valores depois de chamar a funcao:
n1 = -1 e n2 = -2
```

# Ementa

## Listas Ligadas

Definição  
Operações  
Listas duplamente ligadas

1

## Pilhas e filas

Definição  
Operações com pilhas  
Operações com filas

2

## Tabelas de espalhamento

Definição  
Operações  
Otimização

3

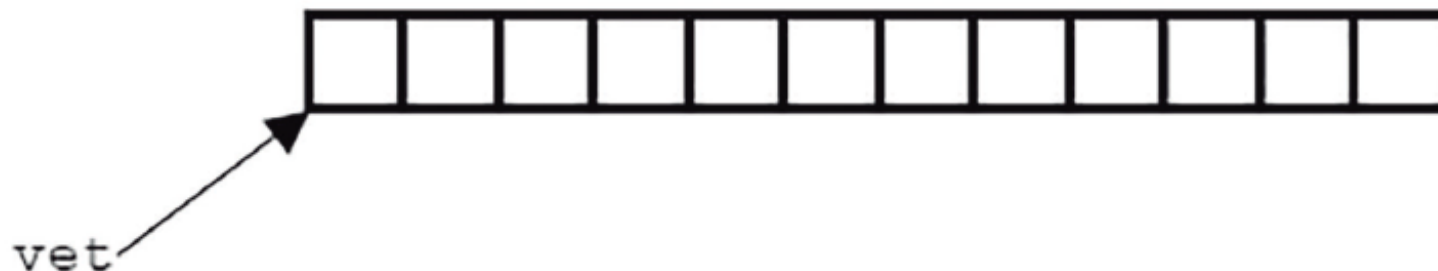
## Armazenamento associativo

Definição  
Mapas com lista  
Mapas com espalhamento

4



Figura 1.1 | Exemplo de vetor



Vetor de 12 posições

```
int vet[12]
```

**E se for necessário aumentar a quantidade de posições de armazenamento do vetor?**

# Alocação dinâmica de memória

---

```
int variável[10];
```

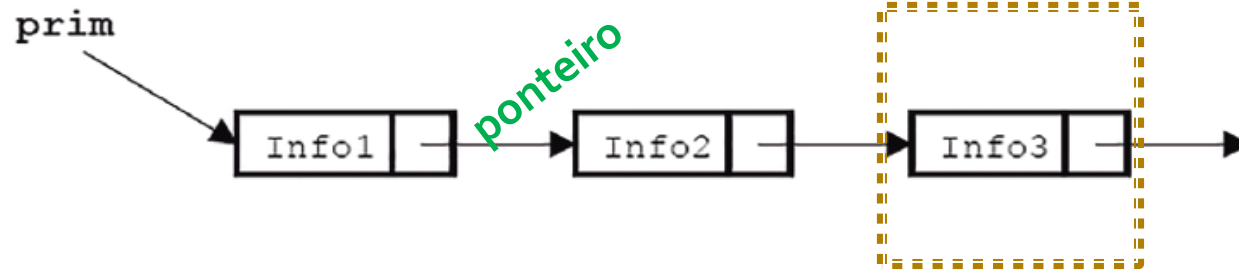
=

```
int* variável;
```

```
variavel = (int*) malloc(10);
```

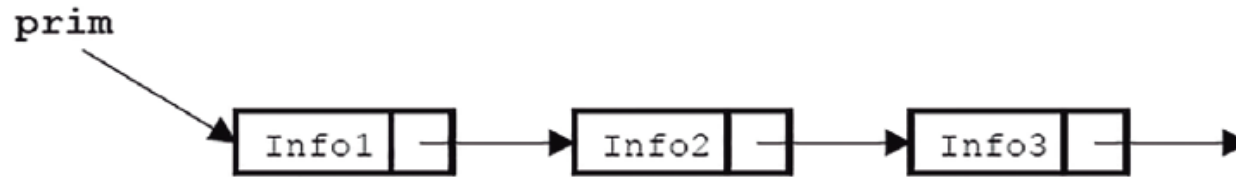
# Listas Ligadas (Encadeadas)

Figura 1.2 | Modelo de lista ligada



E composta de um conjunto de dados dispostos por uma **sequencia de nós**, em que a relação de sucessão desses elementos é determinada por um **ponteiro** que indica a posição do próximo elemento

# Listas Ligadas (Encadeadas)



- Criação ou definição da estrutura de uma lista.
- Inicialização da lista.
- Inserção com base em um endereço como referência.
- Alocação de um endereço de nó para inserção na lista.
- Remoção do nó com base em um endereço como referência.
- Deslocamento do nó removido da lista.

# Listas Ligadas (Encadeadas)

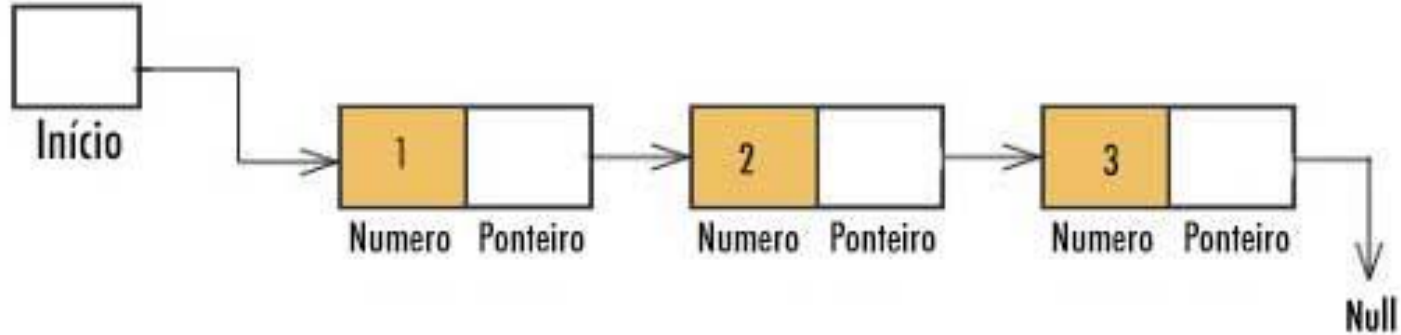
---

Quando uma lista esta sem nós, e definida como **vazia ou nula**, é considerado ponteiro nulo.

Figura 1.3 | Lista vazia com ponteiro nulo

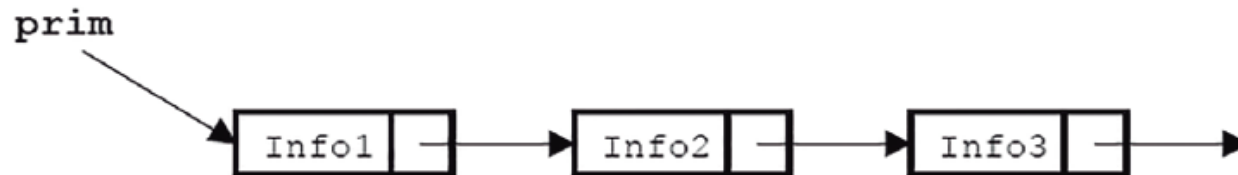


# Listas Ligadas (Encadeadas)



# Listas Ligadas (Encadeadas)

Toda lista precisa ter sua **estrutura definida**, sabendo que cada nó é composto por um conjunto de informações de tipos diferentes e outro de valor inteiro para o ponteiro.



```
struct lista {  
    int info;  
    struct lista* prox;  
};
```

# Listas Ligadas (Encadeadas)



## Exemplificando

Exemplo de declaração para criar uma lista em C:

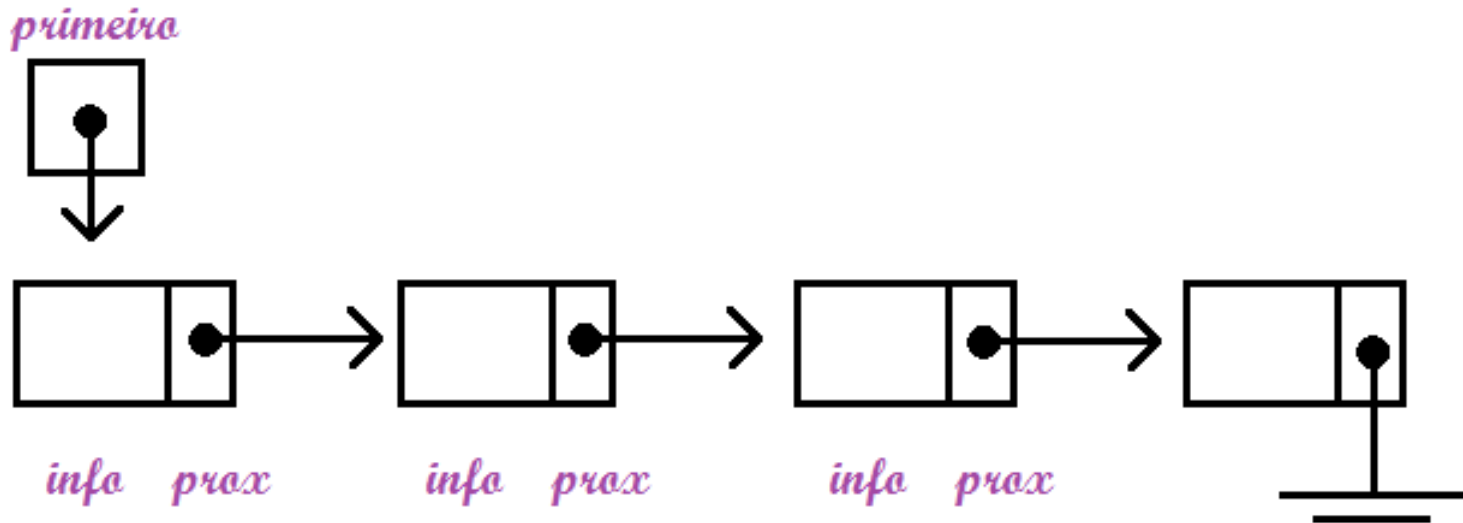
```
/*Cria a estrutura da lista*/  
struct alunos {  
    char nome[25];  
    struct alunos* prox;  
};  
  
};
```

Será criada uma struct (registro) **alunos**:

- Na struct, temos a variável **nome** do tipo char, que será nossa informação;
- Temos outra **struct prox** com ponteiro para a própria struct alunos, para receber o endereço de apontamento da próxima informação.



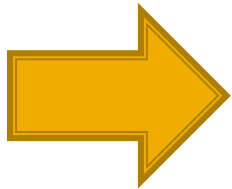
# Listas Ligadas (Encadeadas)



# Listas Ligadas (Encadeadas)

---

Para sabermos o endereço da memória reservada à variável, utiliza-se o operador `&` com o nome de uma variável, enquanto o operador `*`(asterisco), utilizado com a variável do tipo ponteiro, acessa o conteúdo armazenado do endereço de memória, conforme Silva (2007). Temos:



```
int x = 10; /*variável
```

```
int *p; /*ponteiro
```

```
p = &x; /*ponteiro p aponta para o endereço da variável x
```

# Listas Ligadas (Encadeadas)

Figura 1.4 | Exemplo de programa em C com *malloc()* e *sizeof*

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

int main() {

    int *p;
    p=(int *) malloc(sizeof(int));

    if (!p) {
        printf("Erro de memoria insuficiente");
    }else{
        printf("Memoria alocada com sucesso");
    }

    return 0;
}
```

# Structs

```
struct <nome>{  
    <tipo> <nome_da_variavel1>;  
    <tipo> <nome_da_variavel2>;  
    ...  
};
```

```
1.  #include<stdio.h>  
2.  struct automovel{  
3.      char modelo[20];  
4.      int ano;  
5.      float valor;  
6.  };  
7.  main(){  
8.      struct automovel dadosAutomovel1;  
9.      printf("\n Digite o modelo do automovel: ");  
10.     scanf("%s",dadosAutomovel1.modelo);  
11.     printf("\n Digite o ano do automovel: ");  
12.     scanf("%d",&dadosAutomovel1.ano);  
13.     printf("\n Digite o valor do automovel: ");  
14.     scanf("%f",&dadosAutomovel1.valor);  
  
15.     printf("\n Dados atribuidos");  
16.     printf("\n %s",dadosAutomovel1.modelo);  
17.     printf("\n %d",dadosAutomovel1.ano);  
18.     printf("\n %f",dadosAutomovel1.valor);  
19. }
```

struct automovel dadosautomovel [10];

Dadosautomovel[0].modelo = “ Jeep”;

Dadosautomovel[0].ano = 2021;

Dadosautomovel[0].valor = 180000;

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  struct DadosCliente{
5      int codigo;
6      int idade;
7      float salario;
8  };
9  int main(void)
10 {
11     struct DadosCliente vetorClientes[5];
12     int procuracodigo;
13
14     for(int i = 0; i < 3; i++)
15     {
16         printf("\n\n Digite o código do cliente %d:", i+1);
17         scanf("%d", &vetorClientes[i].codigo);
18
19         printf("\n Digite a idade:");
20         scanf("%d", &vetorClientes[i].idade);
21
22         printf("\n Digite o salário:");
23         scanf("%f", &vetorClientes[i].salario);
24
25     }
```

```
27
28 printf("\n\n ***** Digite o código do cliente para recuperar as informacoes:");
29 scanf("%d", &procuracodigo);
30
31 int encontrou = 0;
32 for(int i = 0; i < 3; i++)
33 {
34     if(procuracodigo ==vetorClientes[i].codigo )
35     {
36         printf("\n Idade: %d", vetorClientes[i].idade );
37         printf("\n Salario: %f", vetorClientes[i].salario );
38
39         encontrou = 1;
40
41     }
42 }
43
44 if (encontrou ==0)
45     printf("\n Cliente não cadastrado!!" );
46
47 return 1;
48
49 }
```

# Passagem de parâmetros por referência

```
#include<stdio.h>
void inserir(int a[])
{
    int i=0;
    for(i=0;i<3;i++)
    {
        printf("Digite o valor %d: ",i);
        scanf("%d",&a[i]);
    }
}
void imprimir(int b[])
{
    int i=0;
    for(i=0;i<3;i++)
    {
        printf("\n numeros[%d] = %d",i,2*b[i]);
    }
}

int main()
{
    int numeros[3];
    printf("\n Preenchendo o vetor... \n ");
    inserir(numeros);
    printf("\n Dobro dos valores informados:");
    imprimir(numeros);
    return 0;
}
```

```
Preenchendo o vetor...
Digite o valor 0: 6
Digite o valor 1: 7
Digite o valor 2: 3
```

```
Dobro dos valores informados:
numeros[0] = 12
numeros[1] = 14
numeros[2] = 6
```

# Exercícios 😊



1- Escreva um programa para ler e armazenar as informações de 5 clientes:

- O código do cliente (inteiro);
- A idade;
- O salário.

As informações devem ser armazenadas em structs de maneira apropriada. Em seguida, o programa deverá exibir um menu com as seguintes opções:

1- Exibir clientes
2 - Consultar cliente
3 - Sair do programa

- A opção 1 deverá exibir um relatório contendo todos os dados de todos os clientes cadastrados.
- A opção 2 deverá pedir ao usuário o *código* de um cliente e verificar se tal cliente encontra-se cadastrado, ou seja, se o *código* informado está presente no vetor de códigos. Caso afirmativo, o programa deverá mostrar todos os dados do cliente consultado. Caso contrário, o programa deverá mostrar uma mensagem informando que o cliente não está cadastrado.

2 - Uma Faculdade deseja fazer um levantamento a respeito de seu concurso vestibular e, registrou algumas informações sobre os seus 5 cursos, tais como: código do curso, total de vagas, número de candidatos do sexo masculino e número de candidatos do sexo feminino. Implementar um programa que:

- Leia as informações para cada curso, armazenando em um vetor de código do curso (número inteiro), número de vagas, número de candidatos do sexo masculino e número de candidatos do sexo feminino
- Calcule e imprima, para cada curso, o número de candidatos por vaga.
- O leitor poderá informar o de um curso qualquer e, em seguida, imprimir o número de vagas e o número de candidatos por vaga, juntamente com o código do curso. Caso o código desejado não esteja cadastrado, imprima: 'CURSO INEXISTENTE'.

3 - Escreva um programa em C que faça a leitura da idade e altura de 5 pessoas. O programa deverá encontrar e exibir na tela:

- a. A maior idade;
- b. A altura da pessoa mais nova;
- c. A média aritmética das idades das mulheres;

4 - Faça um programa em C que receba o peso e a altura de 5 pessoas e calcule o Índice de Massa Corporal (IMC) de cada pessoa, que pode ser obtido através da fórmula

$IMC = peso / altura^2$ . O programa deverá exibir a mensagem:

- "Peso Ideal", caso o IMC esteja entre 18.5 e 25;
- ou a mensagem "Você está acima do peso ideal!", caso o IMC esteja acima de 25;
- ou "Você está abaixo do peso ideal", caso o IMC esteja abaixo de 18.5.

5 - Faça um programa que leia um vetor contendo a altura e idade de 10 jogadores de basquete. O programa deverá, também, implementar uma função que receba este vetor como parâmetro e retorne a média de idade dos jogadores que têm mais de 1.90 m de altura.

**1.** Segundo Silva (2007), uma lista ligada, também conhecida como lista encadeada, é um conjunto de dados dispostos por uma sequência de nós, em que a relação de sucessão desses elementos é determinada por um ponteiro que indica a posição do próximo elemento, podendo estar ordenado ou não.

Assinale a alternativa a seguir que apresenta a informação correta quanto à composição de um nó da lista ligada:

- a) Ponteiro para o elemento anterior e uma informação.
- b) Uma informação e um ponteiro para o próximo elemento.
- c) Ponteiro para o próximo elemento e um ponteiro para o elemento anterior.
- d) Ponteiro para o próximo elemento e um dado.
- e) Uma informação e um ponteiro para o elemento anterior.

## 2. Dada a sentença a seguir:

Em uma lista, precisamos \_\_\_\_\_ o tipo de dado no qual foram declarados os \_\_\_\_\_ da lista e, por esse tipo de dados ocupar vários *bytes* na \_\_\_\_\_, precisaremos utilizar a função \_\_\_\_\_, que nos informa quantos *bytes* o tipo de elemento criado terá.

Com base na sentença, assinale a alternativa que apresenta as palavras que completam a frase corretamente:

- a) utilizar, elementos, lista, *sizeof*.
- b) alocar, ponteiros, memória, *sizeof*.
- c) alocar, elementos, memória, *sizeof*.
- d) utilizar, ponteiros, memória, *malloc*.
- e) alocar, elementos, lista, *malloc*.



# Muito Obrigada!

**Prof<sup>a</sup>. Angela Abreu Rosa de Sá, Dr<sup>a</sup>.**

---

*Contato: [angelaabreu@gmail.com](mailto:angelaabreu@gmail.com)*