

CTI UNESP

Profª Kátia Livia Zambon

Adaptação- Apostila de Ling. C da Profª Ariane Scarelli

APOSTILA 1 – Técnicas de Programação

Identifique sua apostila

Nome: _____

Turma: _____

SUMÁRIO

1	INTRODUÇÃO	1
1.1	NÍVEL	1
1.2	PORTABILIDADE	1
1.3	LINGUAGEM ESTRUTURADA	2
2	DESENVOLVIMENTO DE PROGRAMAS	2
2.1	PROCESSO	2
2.2	ESTRUTURA DOS PROGRAMAS	3
3	FUNÇÃO printf	4
3.1	SEQUÊNCIAS DE ESCAPE	5
4	VARIÁVEIS	7
5	FUNÇÃO scanf	9
5.1	CÓDIGOS PARA IMPRESSÃO FORMATADA	14
5.2	ESPECIFICADORES DE COMPRIMENTO E PRECISÃO	14
6	OPERADORES	16
6.1	OPERADORES ARITMÉTICOS	16
6.2	OPERADORES DE INCREMENTO E DECREMENTO	17
6.3	OPERADORES ARITMÉTICOS DE ATRIBUIÇÃO	20
6.4	OPERADORES RELACIONAIS	22
6.5	OPERADORES LÓGICOS	24
6.6	OPERADOR DE ENDEREÇO (&)	26
6.7	OPERADOR CONDICIONAL (?)	26
6.8	OPERADOR DE CONVERSÃO DE TIPO (cast)	28
6.9	ENCADEAMENTO DE EXPRESSÕES	29
7	FLUXOGRAMA	29
7.1	SÍMBOLOS	29
8	COMANDO if	34
9	APÊNDICES	41
	APÊNDICE A – Tipos de Dados em C	41
	APÊNDICE B – Device Driver ANSI.SYS	42

SAUDAÇÃO: Queridos alunos, neste momento não teremos o prazer de nos ver, de dizer bom dia, boa noite, mas sintam-se abraçados por mim. Todos os comentários que eu tiver que fazer estarão destacados desta forma, ok?

Qualquer dúvida podem enviar por e-mail: katia.zambon@unesp.br ou via whatsApp ou pelo ambiente virtual do Classroom.

1 INTRODUÇÃO

A linguagem C foi desenvolvida por Dennis Ritchie em 1972, baseada na linguagem BCPL (Basic Combined Programming Language - Linguagem de Programação Combinada) e na linguagem B, com a finalidade de reescrever o sistema operacional UNIX. No entanto C não está presa ao UNIX ou a qualquer outro sistema operacional ou máquina.

A linguagem C possui a reputação de ser uma linguagem de programação de sistemas porque ela é útil para escrever compiladores e sistemas operacionais, mas pode ser usada também para escrever a maioria dos programas em muitos domínios diferentes.

1.1 NÍVEL

C é chamada de linguagem de médio nível, pois combina características de linguagem de baixo nível e de alto nível.

Baixo nível: possui a funcionalidade e o poder do ASSEMBLY como manipulação de bits, bytes e endereços de memória.

Alto nível: oculta detalhes da arquitetura do computador, aumentando desse modo a eficiência da programação utilizando comandos de fácil entendimento.

1.2 PORTABILIDADE

A portabilidade é uma medida da facilidade com que se pode converter um programa que processa num computador ou sistema operacional para processar em outro computador ou sistema operacional (significa poder adaptar para um dado computador softs escritos para um computador diferente). Durante muito tempo C foi fornecida com o sistema operacional UNIX, mas com a popularidade dos micros, muitas implementações foram feitas e a partir de 1983 o Comitê American National Standards Institute (ANSI) - Instituto Nacional Americano de Padrões - foi reunido com a finalidade de padronizar a linguagem C.

1.3 LINGUAGEM ESTRUTURADA

Uma linguagem estruturada tem como característica a compartimentalização, que é a capacidade da linguagem de seccionar e esconder do resto do programa todas as informações e instruções necessárias somente em um determinado bloco.

2 DESENVOLVIMENTO DE PROGRAMAS

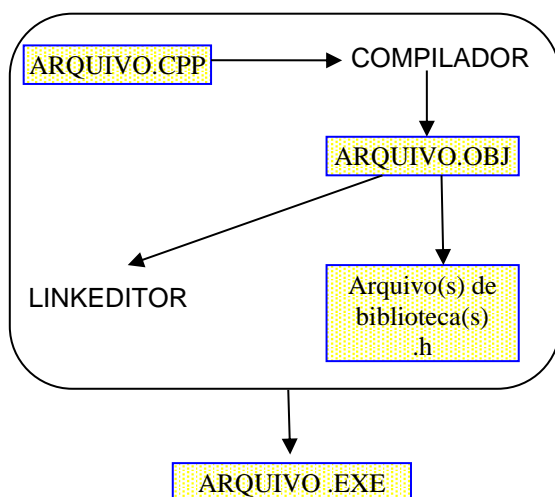
2.1 PROCESSO

O processo de desenvolvimento de programas em C envolve quatro passos:

1. Criação ou edição do programa utilizando um editor
2. Compilação
3. Linkedição
4. Execução

O compilador C é um programa que garante que seu arquivo-fonte não está violando qualquer regra de C. Se o arquivo não contém nenhum erro, o compilador cria um arquivo intermediário com a extensão OBJ. Se o seu programa violar uma das regras de C, o compilador gera um erro de sintaxe.

Processo:



2.2. ESTRUTURA DOS PROGRAMAS

```
nome-da-função(lista-de-argumentos)
    declaração da lista-de-argumentos
    {
        ⇐ Início
        .
        .bloco de declarações
        .
    }
    ⇐ Fim da função
```

Primeiro Programa em C - exemplo:

```
primeiro_prg.cpp
1  /* Nosso primeiro programa de TP */
2  #include <stdio.h>
3  main()
4  {
5      printf("Hello World!!!!");
6  }
```

Onde:

/* */ ⇐ Comentários

main() ⇐ Nome ou local de início do programa;

{ } ⇐ Símbolos agrupadores;

printf ⇐ Comando a ser executado.

Observações:

- **Uso de indentação:** para tornar o programa mais legível utiliza-se um recuo da margem para indicar início e fim das ações;
- **Case-sensitive:** C distingue letras maiúsculas de minúsculas;
- **main():** os parênteses que seguem a palavra *main* são obrigatórios;
- **Uso de ponto-e-vírgula:** deve-se colocar ponto-e-vírgula após cada comando do programa (exceto comandos de repetição: while/do-while/for);
- **Comentários:** a colocação de comentários em C: /* para abrir e */ para fechar, sendo que tudo o que estiver entre os mesmos será ignorado pelo compilador; // indicam que tudo o que estiver à direita das barras será ignorado;
- **Início da execução do programa:** o início lógico é identificado pela função main(), independentemente da sua posição física dentro do programa.

3 FUNÇÃO printf

Definição: a função printf¹ é usada para apresentar informações no vídeo.

Sintaxe: printf("expressão-de-controle", lista-de-argumentos)

onde: **expressão-de-controle:** códigos especiais de controle (sequências de escape) e/ou códigos para impressão formatada.

lista-de-argumentos: separados por vírgula

Segundo Programa em C - exemplo:

```
apostila_exerc9.cpp  [*] exemplo com setlocale.cpp
1  /* Exemplo para utilizar acentuação em Português */
2
3  #include <stdio.h>
4  #include <locale.h>
5  main()
6  {
7      setlocale(LC_ALL, "Portuguese");
8      printf("A vida é bela, filme\n");
9      printf("Qual é a sua música preferida?\n");
10     printf("Geração grátis é melhor");
11 }
```

Responda:

- Qual linha possui comentários e quais são os caracteres que os indicam?
- O que significam as diretrizes das linhas 3 e 4?
- Qual linha é necessária para que possamos utilizar o comando da linha 7?
- Quais linhas são obrigatórias para um programa em linguagem C?

Veja se você entendeu: a função printf é de entrada ou saída de dados?

Se você respondeu saída está correto. Apresenta dados no vídeo



¹ Consulte o APÊNDICE B para tabela com device driver ANSI.SYS.

3.1 SEQUÊNCIAS DE ESCAPE

Sequências de escape são uma série de caracteres usados para manipular a tela e o teclado.

Tabela 1 – Sequência de escapes - *printf*

Sequência de Escape	Resultado
\a	Caractere de beep
\b	Caractere de Backspace
\n	Caractere de mudança de linha
\r	Caractere de retrocesso (1ª coluna)
\t	Caractere de tab horizontal
\\	Caractere de barra invertida
\'	Caractere de aspas simples
\"	Caractere de aspas duplas
\?	Caractere de interrogação

As sequências de escape mais utilizadas são: \n, \t, \a.

Como vocês podem notar na tabela acima existem outras sequências. Nos exercícios de 1 a 6 utilizaremos algumas delas. Tentem fazer os exercícios para entender.

Lembrem-se: Quem é nosso melhor amigo? 😊

EXERCÍCIOS - *printf*

Verifique a execução abaixo e faça os exercícios 1 e 2.

```

C:\Backup_05_10_217\Trabalho_temp\C\2018\apostila_exerc1.exe
if          Se
while       Enquanto
for         Para
before     Antes

-----
Process exited after 0.1522 seconds with return value 0
Pressione qualquer tecla para continuar. . . _

```

Exercício 1: Faça o programa correspondente à execução acima utilizando vários *printf*'s.

Exercício 2: Refaça o exercício anterior com as mesmas configurações (pular as linhas) mas em um único *printf*.

Exercício 3: Observe a sequência de escapes da Tabela 1 e dê o resultado, como ficaria no vídeo se executássemos este código de programa (a proposta aqui é não digitar o programa e sim entender o seu funcionamento):

```
apsotila_exerc3.cpp
1  #include <stdio.h>
2  #include <locale.h>
3  main()
4  {
5      setlocale(LC_ALL, "Portuguese");
6      printf("\nNão pise na grama\" dizia a placa");
7      printf("\nO que é que há velhinho?\n");
8      printf("A banda é de rock\rQual é a Música\?\?");
9  }
```

Exercício 4: Escreva um programa que apresente no vídeo 5 estados do Brasil e suas capitais. Cada informação (estado-capital) deve estar em uma linha.

Exercício 5: Faça um programa que dê a execução do código abaixo. Acrescente mais 3 palavras.

Espanhol	Português
libro	livro
prueba	teste
silla	cadeira
fiesta	festa

Exercício 6: Escreva o resultado das linhas de programação:

- a) printf("Linguagem C\\a Linguagem C\\a");
- b) printf("AB\\b\\bC");
- c) printf("AB\\bC");
- d) printf("Comandos do DOS em C:\\DOS");
- e) printf("\\Fique alerta !\\ gritou o menino");
- f) printf("A\\nB\\nC");
- g) printf("A\\tB\\tC");
- h) printf("XXXAB\\rC");

4 VARIÁVEIS

Uma variável é definida como um espaço de memória reservado para armazenar um tipo de dado que é identificado através de um nome. Essa variável pode ter seu valor alterado durante a execução do programa.

A declaração de variável consiste em:

- Identificar o seu tipo;
- Definir seu nome.

Exemplos: `int var1;`

Obs.: **É obrigatório declarar todas as variáveis utilizadas no programa**

Tabela 2 - Tipos Básicos de Variáveis²

<i>Tipo</i>	<i>Valor Armazenado</i>
int	inteiros de -32768 a + 32767
float	ponto flutuante com 6 ou 7 dígitos de precisão
char	caracteres (ASCII)
double	ponto flutuante com 13 ou 14 dígitos de precisão

Os nomes das variáveis em C podem ser formados por letras, números e sublinhado, sendo o primeiro caractere **obrigatoriamente** uma letra ou o caractere *underline*.

Letras maiúsculas e minúsculas são diferentes (a linguagem C é **case-sensitive**). Os 32 primeiros caracteres são significativos.

As variáveis são importantíssimas para o desenvolvimento de programas, através delas podemos receber valores, fazer cálculos e mostrá-los.

Ao revisar este conteúdo tente responder estas perguntas:

- 1) **O que significa declarar uma variável?**
- 2) **Quais os nomes válidos para variáveis?**
- 3) **O que significa uma linguagem ser case-sensitive?**

² Tabela completa no APÊNDICE A.

Exercício 7: Identifique as variáveis com nomes válidos e o tipo

Nome da variável	Válido	Não válido	Qual o problema?
3num		x	Começa com número
num1	x		
Nota_6			
média			
Robô.virtual			
1_media			
dolar\$			
turma_info			
Teste_prg			
9*6			
Num/4			
Heigth			
pinMode			
Gol			
Time1			
time2			
revisão			
teste-2			
_num			
ana@gmail			
Code.org			
home			
IMC			
Cálculo			
calc_desc			
_result			
x			
Nota1 é igual a nota1?	Sim () Não ()		
result é igual a _result?	Sim () Não ()		
medida é igual a medidax?	Sim () Não ()		

5 FUNÇÃO scanf

Esta função permite ler dados (entrada-*input*) formatados através do teclado. É considerada o complemento da função *printf* (saída – *output*).

A posição de memória onde o valor acessado por *scanf* será armazenado é indicado diretamente pelo endereço da variável (operador &).

Sintaxe:

scanf ("código formatação", endereço da variável);

Código de formatação para *scanf* (que serão utilizados neste ponto do conteúdo):

Código de formatação	Resultado de leitura
%d	número inteiro na base decimal
%f	número em ponto flutuante
%o	número inteiro na base octal
%x	número inteiro na base hexa

Exemplo uso de variável e entrada de dados: faça um programa para ler um valor inteiro, armazenar em uma variável e apresentar o valor digitado e o quadrado do número.

```
1 #include <stdio.h>
2 main()
3 {
4     int num1;
5     printf("Digite um valor: ");
6     scanf("%d",&num1);
7     printf("\nValor digitado: %d",num1);
8     printf("\nQuadrado do valor: %d",num1*num1);
9 }
```

Nós vimos e utilizamos esta função. Preciso que vocês pensem e respondam:

- 1) esta função é para entrada ou saída de dados?
- 2) O que faz o código de formatação "%d". Quando o utilizamos?
- 3) Por que é necessário utilizar o operador "&" na sintaxe desta função?

Outro exemplo com variável do tipo int: Faça um programa que leia (receba) a idade de uma pessoa em anos em uma variável do tipo int. Calcule e apresente (mostre) sua idade em dias (1 ano – 365 dias).

```
apsotila_exerc3.cpp exemplo_scanf_var_int.cpp
1  /* Exemplo utilizando a função scanf */
2  #include <stdio.h>
3
4  main()
5  {
6      int ianos;
7      printf("Digite sua idade em anos: ");
8      scanf("%d", &ianos);
9      int idias = ianos * 365;
10     printf("Sua idade em dias e: %d ", idias);
11 }
```

Responda:

- Quais as variáveis declaradas no programa (nome e tipo)?
- Por que na função *scanf* utilizamos o "&" antes do nome da variável?
- O operador * faz a _____
- O que significa o "%d" no comando *printf* ?
- Os comentários do programa aparecem na execução?
- Quais os caracteres que indicam comentários?
- Por que na função *scanf* utilizamos "%d" e não "%f" para acessar a idade?
- Por que utilizamos o *printf* para mostrar a variável dias e não o *scanf*?

EXERCÍCIOS – com função scanf (int)

Exercício 8: Faça um programa que leia seu número de chamada. Apresente o dobro deste valor.

Exercício 9: Faça um programa que leia um valor inteiro. Calcule e mostre o quadrado e o cubo deste número.

Exercício 10: Observe o programa abaixo e dê como resposta exatamente o que vai aparecer na tela do computador se o usuário digitar 20 e 5.

```
["] Ap1_exerc10.cpp
1  /* exercício 10 da apostila 1 */
2  #include <stdio.h>
3  main()
4  {
5      int num1;
6      printf("Digite o primeiro valor: ");
7      scanf("%d",&num1);
8      int num2;
9      printf("Digite o segundo valor: ");
10     scanf("%d",&num2);
11     printf("\nSoma: %d", num1+num2);
12     printf("\nDiferença: %d", num1-num2);
13     printf("\nMultiplicacao: %d", num1*num2);
14     printf("\nDivisao: %d", num1/num2);
15 }
```

Exercício 11: Faça outra simulação para o programa anterior com valores que você queira e dê o resultado.

Exercício 12: Faça um programa que tenha como entrada (função scanf) a largura e a altura de um cômodo em variáveis do tipo int. Calcule e mostre a área deste cômodo sabendo que o cálculo a ser realizado é largura multiplicado pela altura.

Exercício 13: Faça um programa que leia três valores inteiros, calcule e apresente a média destes valores lidos.

$$MD = (num1+num2+num3)/3$$

Exercício 14: Faça um programa que receba o ano de nascimento de uma pessoa e o ano atual (variáveis do tipo int), calcule e mostre:

- a idade dessa pessoa em anos;
- quantos dias esta pessoa já viveu;
- quantos anos essa pessoa terá em 2050.

Exercício 15: Observe o programa abaixo e dê como resposta exatamente o que vai aparecer na tela do computador. Teste para entrada igual a 15 e depois para entrada igual a 7.

```
xemplo_scanf_var_float.cpp [*] apostila_exerc17.cpp
1 #include <stdio.h>
2 main()
3 {
4     int num;
5     printf("Qual o valor de entrada? ");
6     scanf ("%d",&num);
7     printf("\nO valor de num na operacao 1: %d",num*12);
8     printf("\nO valor de num na operacao 2: %d",num+3*2);
9 }
```

Agora um exemplo com variável do tipo float: faça um programa que leia (receba) o valor da temperatura em Farenheit em uma variável do tipo float. Calcule e apresente (mostre) o correspondente a esta temperatura em graus Celsius.

```
exemplo_scanf_var_float.cpp
1 #include <stdio.h>
2 main()
3 {
4     float ftemp;
5     float ctemp;
6     printf("Informe a temperatura em graus Fahrenheit: ");
7     scanf("%f", &ftemp);
8     ctemp = (ftemp-32)*5/9;
9     printf("Temperatura em graus Celsius: %f\n", ctemp);
10    printf("Temperatura em graus Celsius: %.0f\n", ctemp);
11    printf("Temperatura em graus Celsius: %.1f\n", ctemp);
12    printf("Temperatura em graus Celsius: %.2f\n", ctemp);
13 }
```

```
C:\Backup_05_10_217\Trabalho_temp\C\2018\exemplo_scanf_var_float.exe
Informe a temperatura em graus Fahrenheit: 35
Temperatura em graus Celsius: 1.666667
Temperatura em graus Celsius: 2
Temperatura em graus Celsius: 1.7
Temperatura em graus Celsius: 1.67

-----
Process exited after 14.07 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

Responda:

- Quais as variáveis declaradas no programa (nome e tipo)?
- O que significa o "%f" no comando `scanf`?
- Por que na linha 12 utilizamos "%.2f" e não "%f" para apresentar a variável do tipo float (observe a execução para responder)?
- Na linha 8 temos um cálculo:

```
ctemp= (ftemp-32)*5/9
```

O que significa o sinal de "="?

Qual a ordem de execução das operações?

EXERCÍCIOS – com função scanf (float)

Exercício 16: Faça um programa que leia uma medida em centímetros e converta e mostre o correspondente em polegadas.

1 polegada = 2,54 centímetros

Exercício 17: Faça um programa que leia um valor em reais, calcule e mostre a quantidade em dólares que esta quantia em reais representa.

1	Dólar americano ▼
4,36	Real brasileiro ▼

Exercício 18: Faça um programa que leia uma temperatura em Celsius, calcule e mostre o correspondente em Fahrenheit.

0° Celsius = 32° Fahrenheit

Exercício 19: Escreva um programa que leia o total de horas que uma pessoa utiliza o *whatsapp* durante um dia. Calcule e mostre a **porcentagem** de horas neste aplicativo.

1dia → 24h → 100%

Dica: Para apresentar valores com casas decimais, utilize o recurso do operador de *cast*, que transforma um tipo em outro durante uma operação e a formatação "%.2f" pra apresentar com duas casas decimais.

Exemplo:

```
int var1=80;
int var2=3;
printf("Valor calculado: %.2f",float(var1)/float(var2));
```

Valor calculado: 26,67

Process exited after 0,1677 seconds with return value 0
Pressione qualquer tecla para continuar. . .

Exercício 20: Faça um programa que leia o número total de eleitores de um município, o número de votos brancos, nulos e válidos (variáveis do tipo int). Calcular e escrever o percentual que cada um representa em relação ao total de eleitores.

Exercício 21: Uma empresa resolveu cobrar o uso de games em seu site por horas de acesso. Faça um programa que acesse o número de horas que um jogador quer utilizar, calcule e mostre o valor sabendo que a hora de jogo custa R\$2,59. Apresente o resultado com 2 casas decimais porque é valor monetário na moeda local R\$ (%.2f).

Exercício 22: Elabore um programa que receba (leia com scanf) o valor do raio de uma esfera (tipo float). Calcule e apresente o volume e a área da esfera, baseando-se nas fórmulas dadas. Utilize para *pi* o valor constante de 3.14. Apresente o resultado com 1 casa decimal (%.1f).

Exercício 23: Faça um programa para calcular o custo de uma viagem de automóvel. Para isto receba o total de quilômetros percorridos e o valor $AREA = 4\pi R^2$ pago nos pedágios. Admita que o carro utilize o combustível etanol, que custa R\$ 2.79 e o $VOLUME = \frac{4}{3}\pi R^3$ consumo do veículo é de 8.5 km/l. Apresente o custo total da viagem com duas casas decimais e o símbolo R\$.

Litros = Qtdekm/8.5

Total combustível = Litros * 2.79

Total viagem = Total combustível + total pedágios

Exercício 24: Faça um programa que leia o peso de uma pessoa em kg e a sua altura em metros (variáveis tipo float). Calcular e apresentar o resultado do IMC (Índice de Massa Corpórea) da pessoa. Apresente o resultado com 1 casa decimal (%.1f).

$$IMC = \frac{Peso}{Altura * Altura}$$

Exercício 25: Escreva um programa para informar a largura e o comprimento de um cômodo em metros (variáveis do tipo float). Apresente sua **área** em metros quadrados. Apresentar o resultado com uma casa decimal.

Exercício 26: Faça um programa que calcule para o cômodo do exercício anterior a potência de iluminação necessária para o ambiente. Sabe-se que para cada metro quadrado são necessários 18W de potência. Apresente o resultado sem casa decimal (%.0f).

O item 5.1 e 5.2 é para conhecimento, quando necessitamos fazer alinhamentos na apresentação de dados. Neste momento, não se preocupem. Em momento oportuno, como apresentação de dados em um trabalho, retornaremos a estes tópicos.

5.1 CÓDIGOS PARA IMPRESSÃO FORMATADA

<i>Código de Formatação</i>	<i>Resultado</i>
%c	Apresenta um único caractere
%s	Apresenta uma cadeia de caracteres
%d	Apresenta número inteiro na base decimal
%f	Apresenta número decimal na base decimal
%x ou %X	Apresenta número na base hexadecimal
%o	Apresenta número na base octal
%u	Apresenta número na base decimal sem sinal
%%	Apresenta o caractere %

5.2 ESPECIFICADORES DE COMPRIMENTO E PRECISÃO

Em muitos casos é preciso que os programas exibam as saídas num formato determinado. Isso pode ocorrer especificando um tamanho mínimo de campo para exibir o valor.

Outros exemplos

Valor	Especifi- cador	Saída	Nota
5	%3.3d	005	Deixa 2 zeros p/ indicar precisão p/ 3 díg.
5	%3.2d	05	Deixa espaço p/ preencher o campo; um zero p/ indicar precisão p/ dois dígitos
555	%3.3d	555	Valor contém o número mínimo de dígitos
555	%7.5d	00555	Dois espaços deixados p/ atingir o tamanho do campo; dois zeros p/ indicar a precisão de cinco dígitos
-5	%d	-5	Mostra o valor negativo
3	%04d	0003	Preenche com 3 zeros p/ completar o tamanho do campo

6 OPERADORES

6.1 OPERADORES ARITMÉTICOS

Operador	Significado
=	Atribuição
+	Soma
-	Subtração
*	Multiplicação
/	Divisão
%	Módulo (retorna o resto da divisão entre inteiros)

Exemplos:

$5 / 2 = 2$ (valores tipo int)
 $5 \% 2 = 1$ (valores tipo int)
 $5.0 / 2.0 = 2.5$ (valores tipo float)

EXERCÍCIOS – variáveis numéricas

Exercício 27: Faça um programa no qual sejam declaradas duas variáveis do tipo int, uma variável do tipo float e sejam lidos valores para elas. A seguir, mostre um resultado em cada linha para as operações:

- A soma dos dois valores inteiros;
- O dobro do primeiro valor inteiro somado com o dobro do segundo valor inteiro;
- O valor float multiplicado por 6.5;
- A soma dos três valores.

Exercício 28: Faça um programa que calcule o troco em uma compra. Para isto, insira o valor da compra, o valor de entrada e calcule o troco que o usuário deve receber.

Exercício 29: Elabore um programa que declare uma variável do tipo float que indique uma temperatura em graus Celsius. Transforme-a em Farenheit e Kelvin e apresente os três valores, como no exemplo:

Temperatura em Celsius: 100
 Temperatura em Farenheit: 212
 Temperatura em Kelvin: 373.15

Exercício 30: Para o programa apresentado abaixo, dê os resultados dos comandos *printf* (indique se houve mudança de linha na apresentação dos resultados)

```

1 #include <stdio.h>
2 main()
3 {
4     float fdiv_result = 25.0 / 3.0;;
5     float fmult_result = 3 * 2.5;
6     int idiv_modulo = 25 % 3;
7     float fsoma_result = fdiv_result + fmult_result;
8     float fsub_result = fmult_result - fdiv_result;
9     printf("Divisao float: %.2f\n", fdiv_result);
10    printf("Divisao inteira: %d\n", idiv_modulo);
11    printf("Multiplicacao: %.2f\n", fmult_result);
12    printf("Adicao: %.2f\n", fsoma_result);
13    printf("Subtracao: %.2f\n\n", fsub_result);
14
15    printf("Valor 10 em Decimal %d \n", 10);
16    printf("Valor 10 em Hexadecimal %x \n", 10);
17    printf("Valor 10 em Hexadecimal %X \n", 10);
18    printf("Valor 10 em Octal %o \n", 10);
19    printf("%d * %d * %d = %d\n\n", 2, 3, 5, 2*3*5);
20
21    printf("%f - Sem especificacao\n", 3.3);
22    printf("%.1f - 1 casa decimal\n", 3.3);
23    printf("%.2f - 2 casas decimais e arredondando\n", 12.3456);
24    printf("%.4f - 4 casas decimais completando com zeros\n", 12.345);
25    printf("%.4f - 4 casas decimais completando com espacos\n", 12.3456);
26    printf("Resultados = %d %d", 3+4, 3-4);
27 }

```

Exercício 31: Elabore um programa em que sejam declaradas 2 variáveis do tipo float e a elas atribuídos as notas de 2 provas (os valores deverão estar no intervalo de 0 a 10). Calcule e mostre a média com 1 casa decimal.

Exercício 32: Num jogo de banco imobiliário, uma das opções do jogador é comprar ações de uma empresa petrolífera. Para saber quanto o jogador que "cai" nesta propriedade deve pagar, multiplica o valor da soma dos dados pelo valor da ação da empresa que é de R\$53,60. Faça um programa que leia o valor do dado_1 e do dado_2 (variáveis tipo int), some estes valores e multiplique pelo valor da ação. Mostre o valor monetário a pagar (apresente com duas casas decimais).

6.2 OPERADORES DE INCREMENTO E DECREMENTO

Operador	Significado
++	Incrementa de 1 seu operando
--	Decrementa de 1 seu operando

Os operadores de incremento podem trabalhar nos modos pré-fixado e pós-fixado, onde:

pós-fixado: o operador aparece **após** o nome da variável. O valor da variável é incrementado **após** ser utilizado.

Exemplo:

```

k = 2;
b = k++;
printf("%d %d", k, b);

```

Resultado: k=3 b=2

pré-fixado: o operador aparece **antes** do nome da variável. O valor é incrementado **antes** de ser utilizada.

Exemplo:

```

k = 2;
b = ++k;
printf("k=%d b=%d", k, b);

```

Resultado: k=3 b=3

```
m = 4;
n = 2 * m++;
printf("m=%d n=%d", m, n);
```

Resultado: m=5 n=8

```
m = 4;
n = 2 * ++m;
printf("m=%d n=%d", m, n);
```

Resultado: m=5 n=10

O funcionamento básico do operador de **decremento** é similar ao operador de incremento.

Observações:

- Formatos válidos para os operadores de incremento e decremento:

```
++variavel    variavel++
--variavel    variavel--
```
- Os operadores de incremento e de decremento têm precedência de execução em relação aos operadores aritméticos.
- Esses operadores só podem ser utilizados em variáveis e nunca com expressões ou constantes.

Passo1: identifique a diferença entre os exemplos

Exemplo1	Exemplo2	Exemplo3
a= 7; b = 5; c = ++a * --b	a= 7; b = 5; c = a++ * --b	a= 7; b = 5; c = a++ * b--

Passo2: Sequência de operações realizadas

Exemplo1	Exemplo2	Exemplo3
a= 7; b = 5; a = a + 1; b = b - 1; c = a * b;	a= 7; b = 5; b = b - 1; c = a * b; a = a + 1;	a= 7; b = 5; c = a * b; a = a + 1; b = b - 1;

Passo3: Resultados

Exemplo1	Exemplo2	Exemplo3
a = 8 b = 4 c = 32	a = 8 b = 4 c = 28	a = 8 b = 4 c = 35

EXERCÍCIOS – op. incremento/decremento

Exercício 33: Para as variáveis declaradas e com os valores atribuídos, dê os valores finais que estão sendo solicitados:

33.a) int a = 10;
int b = 5;
int c;
a++;
++b;
c = a + b;

a = _____
b = _____
c = _____

33.b) int a = 7;
int b = 4;
int c;
c = a++ - b;
b - -;

a = _____
b = _____
c = _____

33.c) int a = 2;
int b = 5;
int d;
d = a++ * b;
b = ++d;

a = _____
b = _____
d = _____

33.d) int a = 4;
int b = 6;
int c;
c = ++a * b - -;

a = _____
b = _____
c = _____

33.e) int a = 4;
int b = 6;
int c;
c = - - b * - - a;

a = _____
b = _____
c = _____

33.f) int a = 4;
int c = 2;
int d;
d = 2 * a++ - (-- c);

a = _____
c = _____
d = _____

33.g) int a = 4;
int d;
d = -- a * 4;

a = _____
d = _____

6.3 OPERADORES ARITMÉTICOS DE ATRIBUIÇÃO

Os operadores aritméticos de atribuição apresentados abaixo, agilizam a execução de determinados cálculos já que executam de uma única vez uma atribuição e uma expressão aritmética.

Exemplos:

Operação de Atribuição	Operação Equivalente
a += 5	a = a + 5
a -= 2	a = a - 2
a *= 7	a = a * 7
a /= 3	a = a / 3

```
main()
{
    int valor = 100;
    printf("Valor inicial %d\n", valor);
    valor /= 10;    // valor = valor / 10
    printf("Após a divisao %d\n", valor);
    valor -= 5;    // valor = valor - 5
    printf("Após a subtracao %d\n", valor);
    valor *= 5;    // valor = valor * 5
    printf("Após a multiplicacao %d\n", valor);
}
```

Resolução do exemplo

```
C:\Trabalho\C\teste.exe
Valor inicial 100
Apos a divisao 10
Apos a subtracao 5
Apos a multiplicacao 25
_
```

Outros exemplos:

<i>Operação de Atribuição</i>	<i>Operação Equivalente</i>
$j += 2$	$j = j + 2$
$m *= k + 2$	$m = m * (k + 2)$
$d /= 256$	$d = d / 256$
$e \% = 3$	$e = e \% 3$
$x -= r * 4$	$x = x - (r * 4)$

As expressões com esses operadores produzem um código de máquina mais eficiente já que são mais compactas.

EXERCÍCIOS – operadores de atribuição

Exercício 34: Dê os resultados das operações:

<i>Comando</i>	<i>Resultado</i>
$a = 15 - 2 * 3$	
$a = (15 - 2) * 3$	
$a = 5 * 9 / 15$	
$a = (5 + 2) * 5;$	
$a = 5 + 2 * 5$	
$a = (3 + 5 + 2) * 2$	
$a = (3 + 5) + 2 * 2$	
$a = (3 + 5 * 2) * 2$	

Exercício 35: Fornecidos os valores para as variáveis a, b e c, obter o resultado das expressões e variáveis envolvidas. Considere que em cada item os valores de a, b, c voltam aos valores originais

$a = 2 \quad b = 3 \quad c = 4$

35.a) $c += 10 - a++$

35.b) $b *= 2 + ++c$

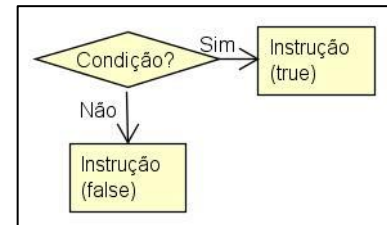
35.c) $c /= 2 + a$

35.d) $b - = 1 * c$

6.4 OPERADORES RELACIONAIS

Os operadores relacionais de C são usados para fazer comparações. São eles:

Operador	Função	Exemplo
>	Maior que	$a > b$
<	Menor que	$a + b < c$
>=	Maior ou igual a	$a \geq b$
<=	Menor ou igual a	$a \leq 0$
==	Igual	$a == b$
!=	Diferente	$!= 0$



Observações:

- A expressão $a = b$ atribui o valor de b para a variável a . A expressão $a == b$ testa se as variáveis a e b são iguais. Um erro comum é usar um único sinal de igual como operador relacional.
- Os operadores aritméticos têm maior precedência que a dos relacionais. Isto significa que serão avaliados antes.

Charada: o que tem de != nos operadores? Podemos dizer que == é == a =?

Há!Há! Sabe aqueles erros que a gente nunca vai cometer???? Mas, acontece?

if (a = b)

if (a == b)

Qual a diferença? Se você não leu as observações anteriores.... deu um "jump" na minha matéria. Volta lá e leia!

Exercício 36: Observe o programa abaixo e dê como resposta o que será impresso.

```

1  #include <stdio.h>
2  main()
3  {
4      int x=10;
5      int y=3;
6      int z=50;
7      if (x > y+z)
8          print("Resultado: %d",x+y+z);
9      else
10         print("Resultado: %d",x*y+z);
11 }
  
```

Exercício 37: Observe o programa abaixo e dê como resposta o que será impresso.

```

1  #include <stdio.h>
2  main()
3  {
4      int x=10;
5      int y=3;
6      int z=50;
7      if (x*y != z*x)
8          print("Equipe 1 vence");
9      else
10         print("Equipe 2 vence");
11 }
  
```


Exercício 38: Utilize seu número de chamada para dar a execução do programa abaixo.

```
1  #include <stdio.h>
2  main()
3  {
4      int num;
5      printf("Entre com o seu número de chamada: ");
6      scanf("%d",&num);
7      int res=num*3;
8      if (res%2 == 0)
9          print("0 que faz o operador ++");
10     else
11         print("0 que faz o operador --");
12 }
```

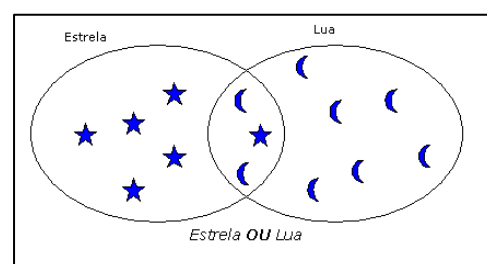
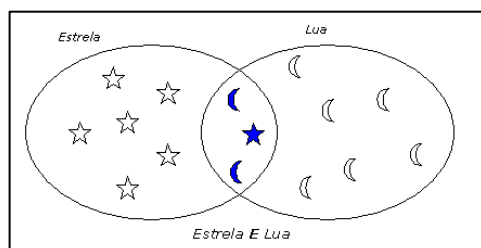
Os operadores do item 6.5 serão detalhados na disciplina Elementos de Lógica e aqui serão utilizados com o comando if para simplificarmos expressões. Tentarei mostrar através dos símbolos gráficos (fluxograma).

Tentem lembrar de duas regras que facilitam:

- **true && true => true**
- **false || false => false**



Veja: http://www.terra.dcc.ufmg.br/material_referencia/mre_PesquisaNaInternet/mre_PesquisaNaInternet.html)



6.5 OPERADORES LÓGICOS

Operador	Função
&&	Lógico E (AND)
	Lógico OU (OR)
!	Lógico de negação

Observem este exemplo: no if da linha 6 se um dos dois valores forem iguais a 1 o resultado será verdadeiro. No if da linha 10 somente será verdadeiro se os dois valores forem maiores que zero. Consegue entender o if da linha 14?

Exemplo:

```

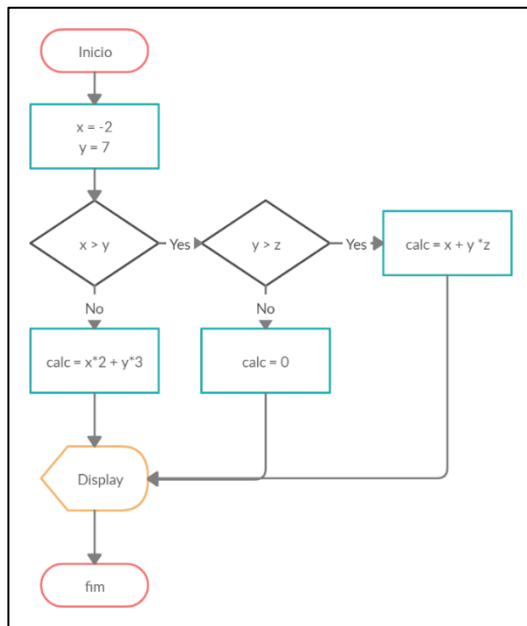
1  #include <stdio.h>
2  main()
3  {
4      int v1=5;
5      int v2=1;
6      if (v1 == 1 || v2 == 1)
7      {
8          printf("um dos valores é 1");
9      }
10     if (v1>0 && v2>0)
11     {
12         printf("\nOs dois valores são positivos");
13     }
14     if (!(v2>1))
15     {
16         printf("\nSegundo valor menor ou igual a 1");
17     }
18 }
```

Exercício 39: Dê os resultados (*true* ou *false*) para as expressões da tabela. Considere que o valor das variáveis são:

a = 12; b = -3; c = 1; d = 0; e = 5;

	Expressão	Resultado
1	(a > b && b > c)	
2	(a > b b > c)	
3	!(a < b)	
4	(c == 1 b < 0)	
5	(a % 2 == 0)	
6	(e % 2 == 0)	
7	(a+b % 2 != 1)	
8	(a++ > 12)	
9	(d*e > 1)	
10	(++a > 12)	

Exercício 40: Percorra o fluxograma que está apresentado abaixo e dê como resposta o resultado da variável "calc".



Exercício 41: Responda qual diversão a pessoa irá. Depois, troque os valores da idade e da altura para obter outros resultados.



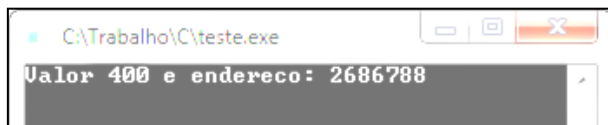
6.6 OPERADOR DE ENDEREÇO (&)

A memória do computador é internamente dividida em unidades de armazenamento chamadas posições de memória. Cada posição de memória é capaz de armazenar uma unidade chamada byte. Todas as variáveis criadas no programa são armazenadas em posições de memória e a elas nos referenciamos através de um nome (ex.: valor).

O operador de endereço permite trabalhar diretamente sobre a posição de memória, assim se utilizarmos uma variável precedida do operador & teremos o endereço do primeiro byte onde a variável está armazenada.

Exemplo:

```
main()
{
    int valor = 400;
    printf("valor: %d e endereco: %u", valor, &valor);
}
```



Pergunta:

- 1) Qual o comando que utilizamos que necessita, em sua sintaxe, do operador de endereço?

6.7 OPERADOR CONDICIONAL (?)

Pessoal, olha que legal. Este operador simplifica o que podemos fazer com o comando if. A resposta na execução será a mesma. Aí o Karl levantou a mão e perguntou: "então para que utilizar o operador "?"

Vamos pensar, a Linguagem C tem várias simplificações de expressões o que fornece um ganho de tempo na execução das mesmas. Então, se substituirmos um if pelo operador interrogação estaremos otimizando nosso algoritmo?

O operador condicional (?) é utilizado para simplificar e resumir expressões condicionais.

Sintaxe:

(expressão) ? exp_verdadeira : exp_falsa

Se expressão for verdadeira, o resultado da operação é exp_verdadeira. Se expressão for falsa, o resultado será exp_falsa.

Quando o compilador C encontra o operador condicional, ele avalia a primeira das três expressões. Se o resultado é verdadeiro, o compilador avalia a segunda expressão, que é, então, o resultado dessa operação. Se a primeira expressão é falsa, então a terceira expressão é avaliada, e seu resultado é a saída da operação.

Exemplo1:

```
1 #include <stdio.h>
2
3 main()
4 {
5     int a=15, b=10;
6     printf ("Max = %d", (a > b) ? a : b);
7     printf ("Min = %d", (a < b) ? a : b);
8 }
```

```
C:\Backup_05_10_217\Trabalho_temp\C\2018\apostila\exemplo_condicional.exe
Max = 15
Min = 10
-----
Process exited after 0.2832 seconds with
Pressione qualquer tecla para continuar
```

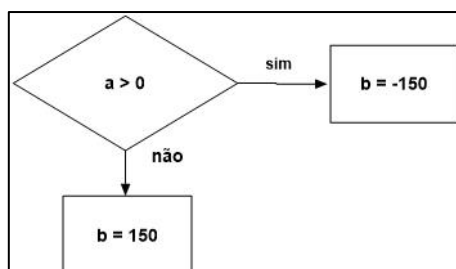
Exemplo2: Observe a mesma condição no if e no operador interrogação:

```
if (a>0)
    b=-150;
else
    b=150;
```

Teremos:

b = a > 0 ? -150 : 150;

No fluxograma:



Exercício 42: Converta as expressões do comando "if" para o operador "?".

- a)

```
if (x > y)
    printf("%d", x-y);
else
    printf("%d", x+y);
```
- b)

```
if (num%2 != 0)
    turma=1;
else
    turma=2;
```
- c)

```
if (temp > 16 && temp < 25)
    printf("outono");
else
    printf("outra estação");
```

Exercício 43: Converta as expressões do operador "?" para o comando if.

- a)

```
(a>10)?printf("\nmaior");printf("\nmenor");
```
- b)

```
(volume >=40)?printf("ligue closed caption");|
printf("nao necessario");
```
- c)

```
int valor=(x%2 != 0)?5:6;
```

6.8 OPERADOR DE CONVERSÃO DE TIPO (cast)

O operador *cast* executa uma conversão forçada de tipo, convertendo uma expressão de um tipo para outro especificado dentro dos parênteses.

Sintaxe: (cast) expressão;

Exemplo: Verifique o exemplo abaixo e a tela de execução. Entenda a diferença dos resultados apresentados.

```
exemplo operador de cast.cpp
1  /* Calculo da media com valores inteiros */
2  #include <stdio.h>
3  main()
4  {
5      int n1 = 12;
6      int n2 = 5;
7      printf ("\nMedia inteira: %d", (n1+n2)/2);
8      printf ("\nMedia float: %.1f", float(n1+n2)/2.0);
9  }
```

```
Media inteira: 8
Media float: 8.5
-----
Process exited after 0.1739 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

Olhe com atenção o exemplo acima. As variáveis $n1$ e $n2$ são de que tipo? int (foi o Enrico que respondeu?). Muito bem, acertou.

Agora olha o cálculo realizado na linha 7 e o resultado que foi impresso: $(n1+n2)/2$ e o resultado é 8. Por que?? Porque neste caso estamos fazendo divisão entre valores inteiros, que gera um resto inteiro também.

Na linha 8 foi utilizado o operador de cast, transformando, somente para este cálculo, as variáveis inteiras em float. O resultado agora foi 8.5.

Ai vocês perguntam quando usar o operador de cast. Ele é usado quando necessitamos de resultados com ponto flutuante, por exemplo.

6.9 ENCADEAMENTO DE EXPRESSÕES

O operador "vírgula" determina uma lista de expressões que deve ser executada sequencialmente. O valor retornado por uma expressão com o operador "vírgula" é sempre dado pela expressão mais à direita.

Exemplo:

$x = (a = 2, a + 3);$

Neste exemplo, a variável x receberá o valor 5.

Este tópico é bem simples e funciona como um simplificar de expressões, que tornam o processamento mais rápido ou mais lento? Será que vocês leram o que eu escrevi?

7 FLUXOGRAMA

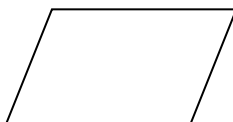
É uma ferramenta que auxilia o desenvolvimento de algoritmos e programas. Isto porque se utiliza a visualização gráfica dos "caminhos" que devem ser seguidos para se resolver um problema. Representado por alguns desenhos geométricos, os quais indicarão os símbolos de entrada, do processamento e da saída de dados.

7.1 SÍMBOLOS



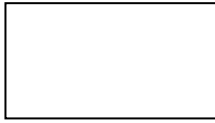
INÍCIO / FIM DO FLUXOGRAMA

Todo fluxograma deve possuir a representação que indica seu Início e Fim



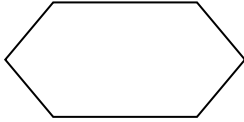
ENTRADA DE DADOS

É utilizada para representar a leitura de uma informação via teclado e armazena esta informação em uma variável de memória



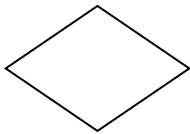
PROCESSAMENTO

É utilizado para representar cálculos e atribuição de valores a variáveis



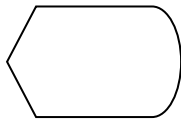
DECLARAÇÃO DE VARIÁVEIS

Todas as variáveis que serão utilizadas no desenvolvimento da resolução do problema devem ser declaradas



DECISÃO

Este símbolo é utilizado para representar uma tomada de decisão, cuja resposta será SIM (verdadeiro) ou NÃO (falso)



SAÍDA DE DADOS

É utilizada para representar a saída de dados



CONECTOR

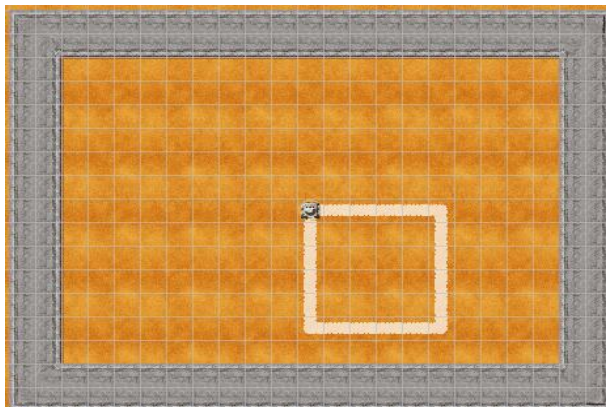
O conector é utilizado quando é preciso particionar o fluxograma. Quando ocorrer a partição, coloca-se uma letra ou número dentro do símbolo de conexão para identificar os pares de ligação.

Exemplo:

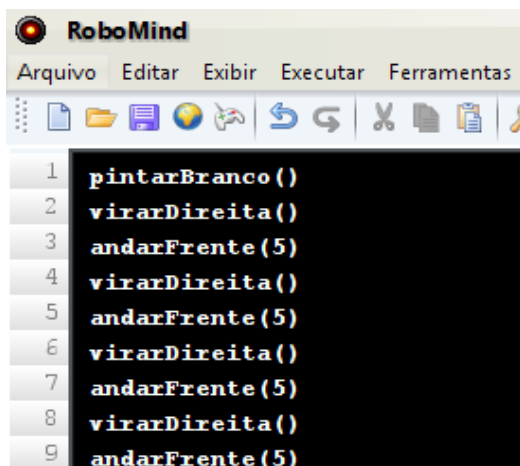
Vamos partir de um exemplo do Robomind para ilustrar o uso desta ferramenta.

Lembram-se do quadrado de dimensão 5?

O desenho proposto para o robô executar é o que aparece na Figura a seguir.



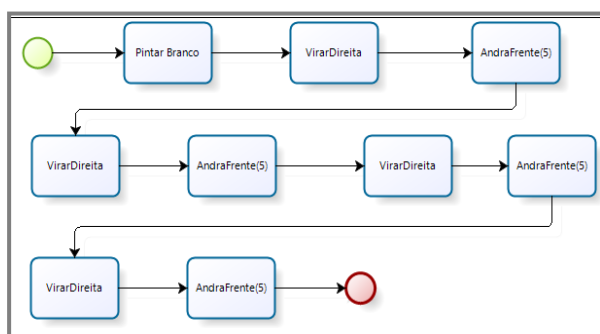
E um possível algoritmo para este problema pode ser dado pela seguinte resolução:



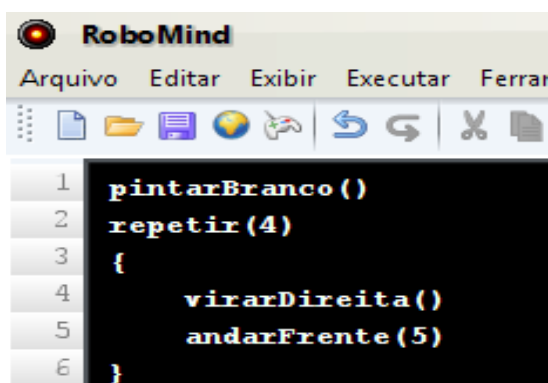
```

1  pintarBranco()
2  virarDireita()
3  andarFrente(5)
4  virarDireita()
5  andarFrente(5)
6  virarDireita()
7  andarFrente(5)
8  virarDireita()
9  andarFrente(5)
    
```

E o fluxograma (aqui com a representação feita no Bizagi Modeler)



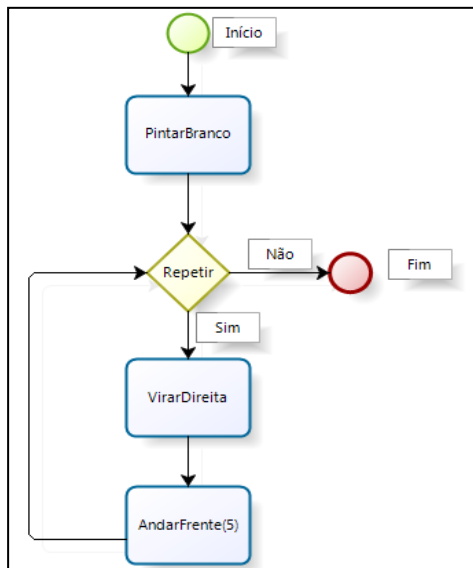
Agora o mesmo problema com outra resolução. Utilizando o "repetir":



```

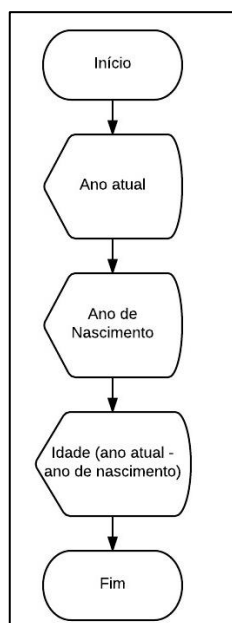
1  pintarBranco()
2  repetir(4)
3  {
4      virarDireita()
5      andarFrente(5)
6  }
    
```

E sua representação de fluxograma:

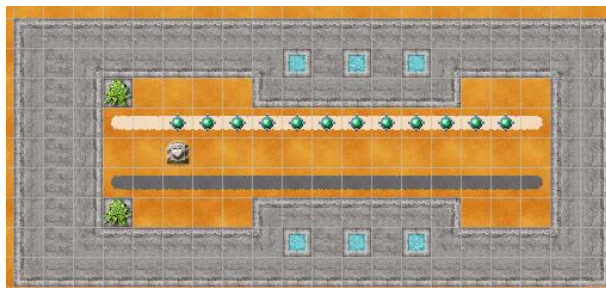


EXERCÍCIOS – algoritmo e fluxograma

Exercício 44: Faça um programa que mostre o ano atual e o ano de seu nascimento. Calcule e mostre sua idade. Observe o fluxograma (feito no LucidChart) que descreve graficamente os passos para a resolução do problema.



Exercício 45: Para resolver o problema abaixo, temos que levar os "beacons" da linha branca para a linha preta (mapa ChangeBelt do Robomind).



Observe o algoritmo e faça o fluxograma.

```

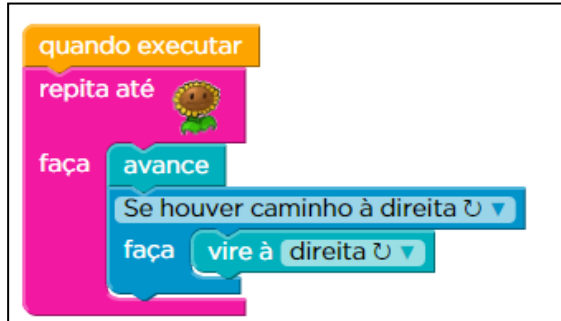
RoboMind
Arquivo  Editar  Exibir  Executar  Fe
1  repetir (12)
2  {
3      pegar ()
4      virarDireita ()
5      virarDireita ()
6      soltar ()
7      virarEsquerda ()
8      andarFrente (1)
9      virarEsquerda ()
10 }
    
```

Exercício 46: Faça um fluxograma para apresentar ler a largura e o comprimento de um cômodo em metros. Pode utilizar números com ponto decimal. Apresente sua área em metros quadrados.

Exercício 47: Observe a atividade 14 da Fase 2 do Code no qual o objetivo é que o "marciano" chegue até à flor e não seja devorado pelas plantas carnívoras:



A resolução proposta neste caso está apresentada abaixo. **Faça o fluxograma** correspondente utilizando o símbolo de decisão para que ele possa virar na direção correta. Verifique os exemplos anteriores.



8 COMANDO if

Utilizado na tomada de decisão baseado no resultado verdadeiro ou falso da expressão.

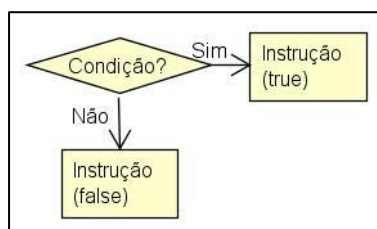
Sintaxe: if (condição de teste)

Instrução(true);

else

instrução(false);

Representação no fluxograma:



Considerando as tarefas desenvolvidas no **Code.org**, veja o exemplo abaixo:

Repete comigo:

"Depois do if NÃO tem ponto-e-vírgula"

"Depois do if NÃO tem ponto-e-vírgula"

"Depois do if NÃO tem ponto-e-vírgula"

"Depois do if NÃO tem ponto-e-vírgula"

.....

"Depois do if NÃO tem ponto-e-vírgula"

Umas 100 vezes?? Tipo um "mantra"?

Code.org: Fase 9 – Quebra-cabeça



SE há um buraco
faça encher 1

SE há uma pilha
faça remover 1

Exemplo1: if sem else

```
1 #include <stdio.h>
2 main()
3 {
4     int valor = 120;
5     if (valor > 100)
6         printf("Este valor e maior que cem");
7 }
```

Exemplo2: if - else

```
1 #include <stdio.h>
2 main()
3 {
4     float fQtde;
5     printf("Digite quantos litros de água: ");
6     scanf("%f", &fQtde);
7     if (fQtde > 500)
8         printf("Conta alta, economizar!!");
9     else
10        printf("Consciente, continue assim");
11 }
```

Exemplo3: if-else

```
1 #include <stdio.h>
2 main()
3 {
4     int iTeste = 5;
5     if (iTeste > 0)
6     {
7         int iA = 10;
8         iA *= 8;
9         printf("valor final de iA = %d", iA);
10    }
11    else
12    {
13        int iA = 20;
14        iA *= -2;
15        printf("valor final de iA = %d", iA);
16    }
17 }
```

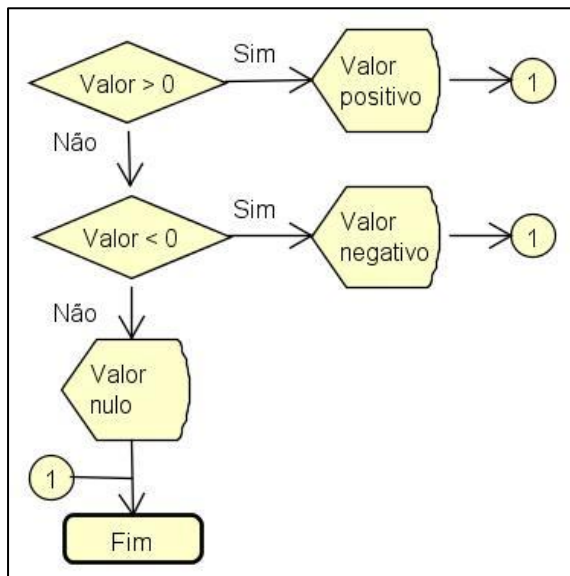
Exemplo4: if-else-if

```

1 #include <stdio.h>
2 main()
3 {
4     int valor;
5     scanf("%d",&valor);
6     if (valor > 0 )
7         printf("Número positivo");
8     else
9         if (valor < 0 )
10            printf("Número negativo");
11        else
12            printf("Valor nulo");
13 }

```

Representação no fluxograma:



Exemplo5: if-else-if

```

1 #include <stdio.h>
2 main()
3 {
4     int valor = 50;
5     if (valor <= 100)
6     {
7         if (valor < 10)
8             printf("valor menor que 10");
9         }
10    else
11        printf("valor maior que 100");
12 }

```

Exemplo6: if-else-if

```

1 #include <stdio.h>
2 main()
3 {
4     int andar = 3;
5     if (andar == 1)
6         printf("Primeiro andar");
7     else if (andar == 2)
8         printf("Segundo andar");
9     else if (andar == 3)
10        printf("Terceiro andar");
11    else
12        printf("Andar invalido");
13 }

```

EXERCÍCIOS – comando if

Exercício 48: Faça um fluxograma e um programa para ler um valor inteiro e apresentar uma mensagem que indique se o número é maior que 100 ou menor que 100.

Exercício 49: Faça um programa para ler 2 valores inteiros (considere que não serão informados valores iguais) e escrever o maior deles.

Exercício 50: Faça um programa para ler o ano de nascimento de uma pessoa, calcular sua idade e escrever uma mensagem que diga se ela poderá ou não votar nas eleições deste ano.

Exercício 51: Faça um programa que leia um número inteiro e apresente uma mensagem indicando se este número é par ou ímpar.

Dica: Utilize o operador % que retorna o resto da divisão entre inteiros e verifique se o resto é zero.

Exercício 52: Faça um programa que leia o placar de um jogo de futebol (os gols de cada time, um valor em cada variável do tipo int). Informe se houve empate ou se a vitória foi do 1º ou do 2º time.

Exercício 53: João Pescador comprou um microcomputador para controlar o rendimento diário de seu trabalho. Toda vez que ele traz um peso de peixes maior que o estabelecido pelo regulamento de pesca (50 quilos) deve pagar uma multa de R\$ 4,00 por quilo excedente. João precisa de um programa que leia o peso de peixes e verifique se há excesso, mostrando a quantidade de quilos excedidos e o valor da multa que João deverá pagar. Caso contrário, mostrar uma mensagem de que não houve excesso.

Exercício 54: Faça um programa que leia a quantidade de energia elétrica consumida em KW no decorrer de um mês em uma residência. Considerando o preço do KW de R\$ 0,50, determinar o valor a ser pago, sabendo

que se o valor for inferior a R\$ 30,00 o consumidor recebe um desconto de 10% em sua conta, caso contrário, recebe um desconto de 5% (comando if). Apresentar o valor a ser pago.

Exercício 55: Escreva um programa para transformar a temperatura fornecida em Fahrenheit para Celsius. $C = 5 / 9 * (F - 32)$

- se temp. em Celsius for menor ou igual a zero, imprimir "Frio ártico !";
- se temp. em Celsius de 01 a 12 graus, imprimir "Muito frio !";
- se temp. em Celsius de 13 a 23 graus, imprimir "Clima ameno !";
- se temp. em Celsius de 24 a 32 graus, imprimir "Calor !";
- se temp. em Celsius de 35 a 40 graus, imprimir "Calor tórrido !";
- Qualquer outro valor, imprimir "Sem registro !";

Exercício 56: Em uma competição de saltos ornamentais, 6 (seis) juízes informam notas reais variando de 0 a 10. A nota final do atleta deve excluir a maior e a menor nota dos juízes e é composta pela soma das quatro demais notas. Faça um programa que lê do usuário as seis notas dos juízes e informa a nota final do atleta (a soma das notas excluindo a menor e a maior delas).

Exercício 57: Escreva um programa que leia a idade de 2 homens e 2 mulheres (considere que a idade dos homens será sempre diferente, assim como das mulheres). Calcule e escreva a soma das idades do homem mais velho com a mulher mais nova, e o produto das idades do homem mais novo com a mulher mais velha.

Exercício 58: Leia a velocidade máxima permitida em uma avenida e a velocidade com que o motorista estava dirigindo nela e calcule e apresente a multa que uma pessoa vai receber, sabendo que são pagos:

- 50 reais se o motorista estiver ultrapassar em até 10km/h a velocidade permitida (ex.: velocidade máxima: 50km/h; motorista a 60km/h ou a 56km/h);
- 100 reais, se o motorista ultrapassar de 11 a 30 km/h a velocidade permitida.
- 200 reais, se estiver acima de 31km/h da velocidade permitida.

Exercício 59: Escreva um programa que leia o valor de 3 ângulos de um triângulo e escreva se o triângulo é acutângulo, retângulo ou obtusângulo. Observação:

- Triângulo retângulo: possui um ângulo reto (90 graus)
- Triângulo obtusângulo: possui um ângulo obtuso (ângulo maior que 90 graus)
- Triângulo acutângulo: possui 3 ângulos agudos (ângulo menor que 90 graus)

Exercício 60: Elaborar um programa para ler os três lados de um triângulo, verificar se formam um triângulo (cada lado tem que ser menor que a soma dos outros dois) e identificar o tipo de triângulo: equilátero => 3 lados iguais, isósceles => 2 lados iguais, escaleno => 3 lados diferentes.

Exercício 61: Faça um programa que auxilie o cálculo para o aumento no salário de jogadores de futebol. Leia o salário atual, calcule o aumento e mostre o novo salário.

Salário atual	Aumento
R\$ 0,00 – R\$ 2.300,00	20%
R\$ 2.300,01 – R\$ 4.500,00	15%
R\$ 4.500,01 – R\$ 7.000,00	10%
Acima de R\$ 7.000,00	5%

Exercício 62: Faça um programa que receba o código de um produto e o classifique de acordo com a tabela abaixo:

Código	Classificação
1	Alimento não perecível
2 a 4	Alimento perecível
5 a 6	Vestuário
7 a 9	Higiene pessoal
10 a 15	Utensílios e utilidades domésticas
Qualquer outro código	inválido

Exercício 63: Em uma passagem de nível, a cancela é fechada automaticamente quando o trem está a 100 m do início do cruzamento. O trem, de comprimento 200 m, move-se com uma velocidade constante de 36 km/h. Assim que o último vagão passa pela final do cruzamento, a cancela se abre liberando o tráfego de veículos. Considerando que a rua tem largura de 20 m, o tempo que o trânsito fica retido desde o início do fechamento da cancela até o início da sua abertura é em segundos:

- 1) 32 2) 36
3) 44 4) 54 5) 60

Escreva um programa que apresente o enunciado do problema, acesse a resposta do usuário e com o comando “if” verifique se está correta e dê uma mensagem para o usuário. Se ele errar a resposta, mostre a correta.

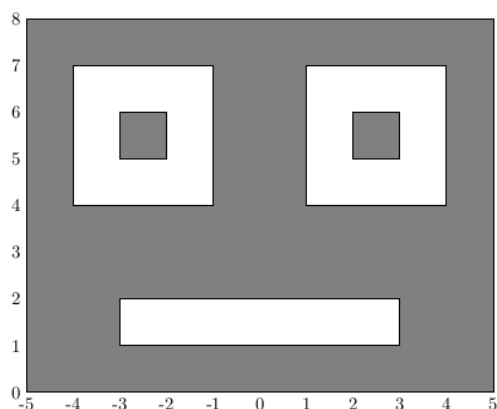
Exercício 64: Faça um programa que entre com dois valores inteiros x e y. Estes valores de entrada são as coordenadas de uma mosca que pousa na carinha ou não.

Observando:

- a coordenada x vai de -5 a 5, correto?
- A coordenada y vai de 0 a 8, certo?

Então, se x e y estiverem fora destes parâmetros, a mosca não pousou na cara. Mas, ela pode pousar na cara, no olho direito, na pupila direita, no olho esquerdo, na pupila esquerda ou na boca.

O seu programa deve identificar onde a mosca pousou considerando as coordenadas que identificam as partes da carinha.



Dica: utilize os operadores relacionais para facilitar o processo

REFERÊNCIAS

CASTRO, André L. F. **Apostila C**. [200-].

CODE.org. Disponível em: <https://www.code.org/> Acesso em: 08 de fevereiro de 2019.

JAMSA, Kris. **Microsoft C, dicas, segredos e truques**. São Paulo: Makron Books, 1992, 701p.

MIZRAHI, Victorine V. **Treinamento em linguagem C, curso completo**. São Paulo: McGraw-Hill, 1990. (Módulo 1). 241p.

_____. _____. São Paulo: McGraw-Hill, 1990. (Módulo 2). 273p.

RODOMIND.net. Disponível em: <https://www.robomind.net/> Acesso em: 08 de fevereiro de 2019.

"Everybody in this country should learn to program because it teaches you how to think" — Steve Jobs

9 APÊNDICES

APÊNDICE A – Tipos de Dados em C

O C tem 5 tipos básicos: **char**, **int**, **float**, **double**, **void**. O **double** é o ponto flutuante duplo e pode ser visto como um ponto flutuante com muito mais precisão. O **void** é um tipo especial que significa vazio em inglês.

Tipo	Número de bits	Intervalo	
		Início	Fim
char	8	-128	127
unsigned char	8	0	255
signed char	8	-128	127
int	16	-32.768	32.767
unsigned int	16	0	65.535
signed int	16	-32.768	32.767
short int	16	-32.768	32.767
unsigned short int	16	0	65.535
signed short int	16	-32.768	32.767
long int	32	-2.147.483.648	2.147.483.647
signed long int	32	-2.147.483.648	2.147.483.647
unsigned long int	32	0	4.294.967.295
float	32	3,4E-38	3,4E+38
double	64	1,7E-308	1,7E+308
long double	80	3,4E-4932	3,4E+4932

O tipo **long double** é o tipo de ponto flutuante com maior precisão. É importante observar que os intervalos de ponto flutuante, na tabela acima, estão indicados em faixa de expoente, mas os números podem assumir valores tanto positivos quanto negativos.

APÊNDICE B – Device Driver ANSI.SYS

Sob o DOS, o *device driver* (controlador de dispositivo) ANSI.SYS fornece expansão das capacidades da tela e teclado, utilizando um código padronizado pelo *American National Standards Institute* (ANSI).

O *driver* ANSI.SYS intercepta toda saída na tela e a entrada do teclado e, se algum código interceptado for uma sequência especial de caracteres que identificam um comando para o *driver*, ele é passado para sua parte de processamento de comando.

Os comandos para o *driver* ANSI.SYS são identificados por uma sequência especial de caracteres inicializada por ESCAPE (1B em hexa) + [.

O *driver* ANSI.SYS deve estar instalado no CONFIG.SYS.

Exemplo: "\x1B[2J"

Onde: \x ⇒ informa que os 2 próximos caracteres formam um valor hexadecimal.

1B[⇒ identifica um código ANSI.

2J ⇒ identifica a ação (limpa a tela e posiciona o cursor no canto superior esquerdo do vídeo).

```
main()
{
    printf("\x1B[2J");
}
```

Manipulando o Vídeo

Sequência de Escape	Função	Exemplo
"\x1B[2J"	Limpa tela e posiciona no canto superior esquerdo.	printf("\x1B[2J");
"\x1B[#;#H"	Define linha e coluna do cursor	printf("\x1B[5;10H");
"\x1B[#A"	Move o cursor # linhas acima	printf("\x1B[3A");
"\x1B[#B"	Move o cursor # linhas abaixo	printf("\x1B[10B");

"\x1B[#C"	Move o cursor # colunas à direita	printf("\x1B[6C");
"\x1B[#D"	Move o cursor # colunas à esquerda	printf("\x1B[11D");
"\x1B[s"	Salva posição atual do cursor	printf("\x1B[s");
"\x1B[u"	Retorna o cursor à posição anterior	printf("\x1B[u");
"\x1B[K"	Limpa até o final da linha atual	printf("\x1B[K");
"\x1B[#m"	Define cor de primeiro plano e fundo	printf("\x1B[41m");

Utilizando Cores

Valor	Cor de Primeiro Plano
30	Preto
31	Vermelho
32	Verde
33	Laranja
34	Azul
35	Magenta
36	Ciano
37	Branco
Valor	Cor de Fundo
40	Preto
41	Vermelho
42	Verde
43	Laranja

44	Azul
45	Magenta
46	Ciano
47	Branco

Exemplo:

```
main()
{
    printf("\x1B[43mFUNDO LARANJA");
    printf("\x1B[31mVERMELHO\x1B[37mBRANCO\x1B[34mAZUL");
}
```

Onde: o **m** minúsculo é a sequência de escape que diz ao driver ANSI que o número precedente é a cor escolhida para ser mostrada no vídeo, como primeiro plano ou fundo, dependendo do valor.

Como fazer para ativar o *device driver* ANSI.SYS no *prompt* de comando do Windows XP, Vista.

Passos:

- 1 Abra (edite) o arquivo CONFIG.NT (que substituiu o antigo CONFIG.SYS), localizado em c:\windows\system32
- 2 Insira a seguinte linha:
device= c:\windows\system32\ansi.sys
(ou device= %SystemRoot%\system32\ansi.sys)
- 3 Salve o arquivo com as alterações
- 4 Reinicie o computador

Na próxima vez que executar o prompt de comando do Windows (CMD) ou o atalho do Dev C++ O driver ANSI.SYS interceptará toda saída na tela e entrada do teclado