[**http://tinyurl.com/ppbadyn**](http://tinyurl.com/ppbadyn)

[**http://tinyurl.com/n6aqh44**](http://tinyurl.com/n6aqh44) **Exercícios de Revisão**

[**http://tinyurl.com/n3wbqgb**](http://tinyurl.com/n3wbqgb) **- Correção da Prova AV1**

[**http://tinyurl.com/lduh2qd**](http://tinyurl.com/lduh2qd) **- Exercícios para apresentar**

**Notas**

[**http://tinyurl.com/mxyouwp**](http://tinyurl.com/mxyouwp) **//Vergueiro Notas AV2**

[**http://goo.gl/tcsjJ2**](http://goo.gl/tcsjJ2) **// Santo Amaro Notas AV2**

[**http://goo.gl/r8Ruu1**](http://goo.gl/r8Ruu1) **// Vila Prudente Exercícios**

**IDE’s (Ambientes de Desenvolvimento Integrado)**

Para PC Windows: **Dev C++/Code Blocks/Visual Studio**

Para Android: **c4droid**

Bibliografia

Livros sugeridos

* Damas, Luís. Linguagem C, Rio de Janeiro:LTC, 2013.
* Schildt, Hebert. C completo e total, Makron Books.
* Manzano, José Augusto. Linguagem C, Estudo Dirigido, São Paulo: Érica
* Kerninghan, Brian; Ritchie, Dennis. Linguagem C padrão ANSI, Rio de Janeiro: Campus.
* Treinamento em Linguagem C: Módulo 1 e 2; Mizhari, Viviane Victorine..Pearson.
* Brod, Cesar. Aprenda a Programar. Ed. Novatec

**Algoritmos e Lógica de Programação**

Quando desejamos ensinar alguma tarefa mais complexa à outra pessoa, normalmente, relacionamos passo a passo as instruções que esta pessoa deve cumprir para atingir o objetivo.

Mas qualquer tarefa pode ser desmembrada em instruções passo a passo de modo a atingir um objetivo final.

Por exemplo, um procedimento para trocar o pneu furado do carro poderia ficar assim:

|  |
| --- |
| Pegue todas as ferramentas necessárias;  Se o estepe estiver em boas condições então  continue;  Senão  vá ao borracheiro;  Solte os parafusos da roda;  Levante o carro com auxílio do macaco;  Tire os parafusos e a roda;  Coloque o estepe;  Aperte os parafusos da roda;  Abaixe o carro;  Aperte, definitivamente, os parafusos;  Guarde as ferramentas e o pneu furado. |

A esse conjunto de instruções podemos denominar algoritmo. A palavra algoritmo é uma derivação de Al-Khwarizmi, que é o nome de um matemático árabe precursor da álgebra.

O site

<http://plastelina.net/>

Ele possui jogos de lógica. Visite-o, resolva os problemas propostos e escreva um algoritmo descrito para documentar todas as etapas utilizadas para solucionar o problema.

Definições de algoritmo:

a) Conjunto das regras e procedimentos lógicos perfeitamente definidos que levam à solução de um problema em um número finito de etapas **(Dicionário Houaiss)**.

b) Um conjunto de instruções objetivas e finitas, com encadeamento lógico entre elas, para a realização de uma tarefa.

Tipos de algoritmos

**Narrativo** – É o algoritmo escrito diretamente na linguagem natural. Por exemplo, o algoritmo descrito acima. Nele narramos as instruções que devem ser realizadas.

**Fluxograma (ou diagrama de blocos)** – É uma representação universal, ela utiliza-se de símbolos gráficos específicos que representam determinadas ações. E o conjunto de símbolos gráficos, conectados por setas de fluxo, representam um algoritmo.

**Português Estruturado** – É uma forma de algoritmo intermediário entre a linguagem descritiva (natural) e a linguagem de programação. Apresenta o conjunto de instruções estruturadas, semelhante a um programa de computador. Porém, as instruções são escritas em português e não estão codificadas.

**Linguagem de programação** – Código composto de um número conhecido de palavras ou de expressões que podem ser encadeadas obedecendo a um conjunto de regras fixas denominado sintaxe, capaz de gerar uma série de instruções passíveis de ser interpretadas por um computador, formando então um programa (Dicionário Houaiss).

Exemplos de linguagens de programação: Java, C, C#, C++, Pascal, Delphi entre outras.

Exemplos de representação de algoritmos

principal(){

**real a;**

**escreva(“Digite um número”)**

**leia(a)**

**escreva(“Você digitou ”, a)**

}

Em C

#include<stdlib.h>

#include<stdio.h>

int main(){

system(“title Exemplo1”);

system(“color 1E”);

float a;

printf(“Digite um numero\n”);

scanf(“%f”,&a);

printf(“Voce digitou %.1f\n”,a);

system(“pause”);

return 0;

}

Todos os algoritmos têm três aspectos fundamentais. São eles: entrada de dados, processo e saída. Com efeito, para o primeiro algoritmo aqui mencionado (procedimento da troca de pneu furado) temos que a entrada é o pneu furado e as ferramentas utilizadas, o processo é a troca, efetivamente, do pneu furado e a saída é o carro em condições de uso.

**Linguagem de Programação**

A linguagem de programação é o resultado final da idealização de um algoritmo, para efetivamente programá-lo em um computador. A linguagem que utilizaremos, para isso, é a linguagem C.

Ela é uma linguagem **compilada**, o que equivale a dizer que é uma linguagem nativa da máquina. Portanto, após escrever o programa (código-fonte) em um editor, devemos compilar, ou seja, devemos convertê-la para a **linguagem de máquina**. Que é a linguagem de 0’s e 1’s, são essas as informações que o computador realmente entende.

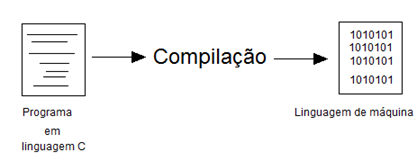


Figura 2 – Código-fonte compilado

Porém, existem linguagens que são interpretadas, como o Java e o C#, por exemplo. A diferença entre a linguagem interpretada e a linguagem compilada está relacionada com o procedimento na sua execução.

Após escrevemos um programa em Java, desejamos executá-lo e para isso, faz-se necessário a máquina virtual, que é um **interpretador**. É através dele que o programa escrito será convertido em linguagem de máquina. Mas essa conversão é realizada linha por linha até o comando final do programa, no momento da execução. De modo que toda vez que pedirmos para executar aquele programa, ele será novamente interpretado. Não ficando, portanto, nativo da máquina, ou seja, é independente de plataforma.

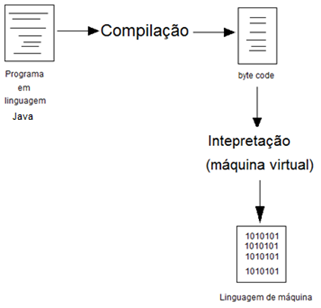


Figura 3 – Código-fonte interpretado

**Linguagem de Programação**

**A linguagem C**

**Primeiro Exemplo de Linguagem C**

#include<stdio.h>

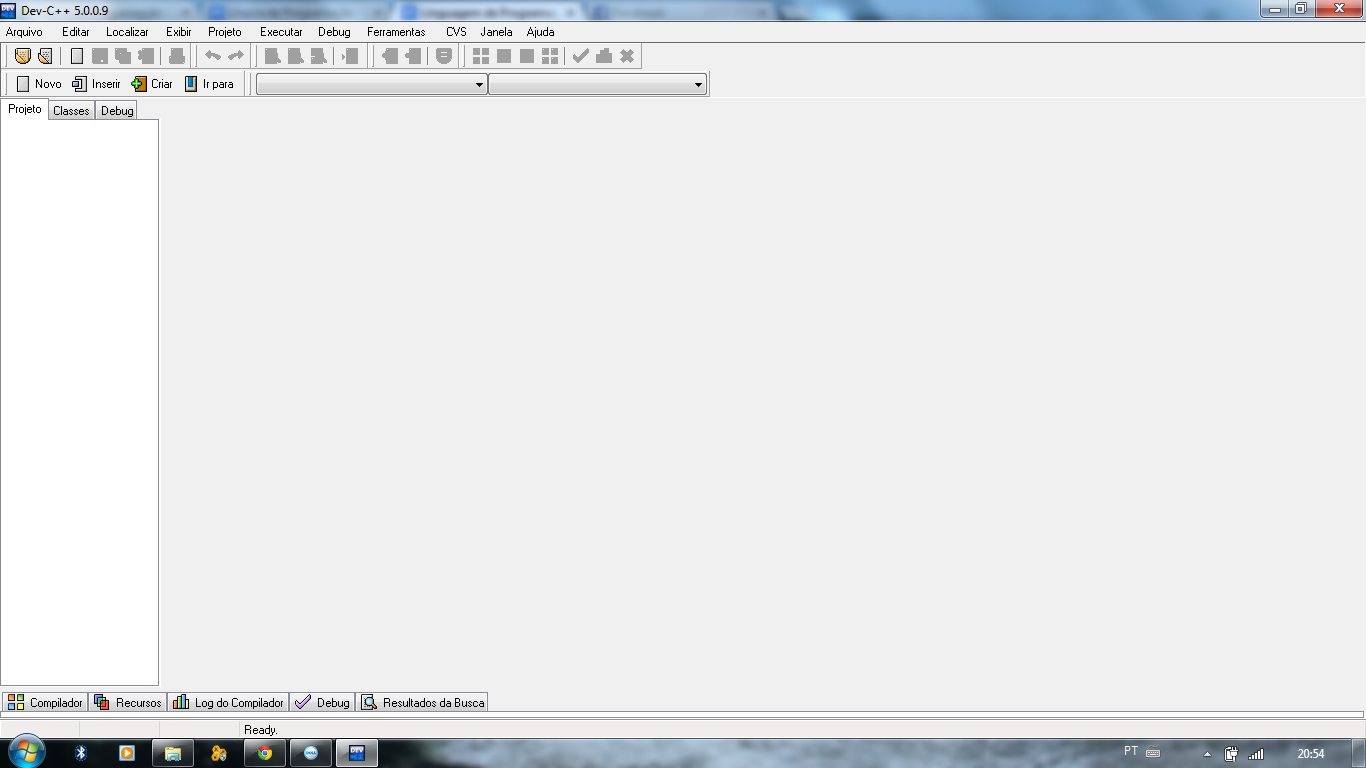
main(){

printf(“Primeiro Exemplo\n”); //escreva

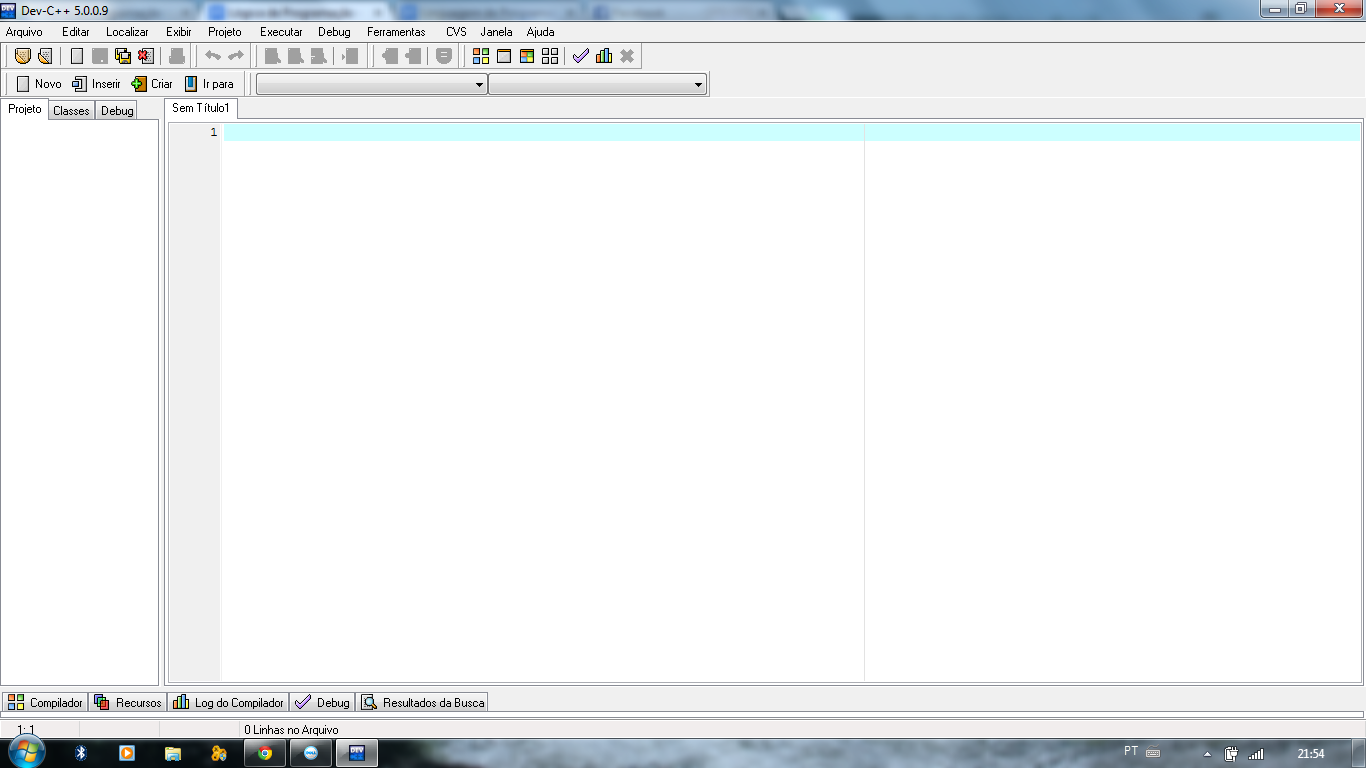
system(“pause”);

}

**No Dev C++**



**Pressione Ctrl+n**



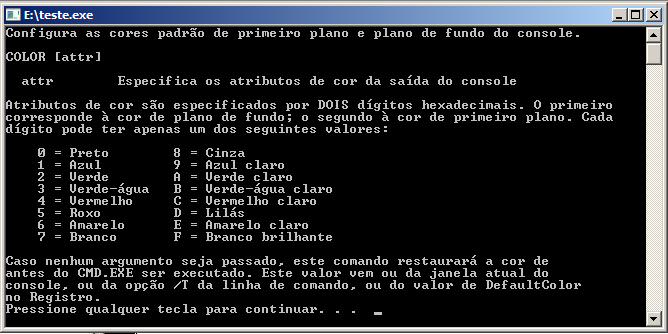
Salve o arquivo assim:

Nome do arquivo: **Exemplo1**

Tipo: **.C**

**Para compilar e executar o arquivo**

**F9**



**Tabela de valores para o comando color**

**/\*Primeiro Exemplo**

**Autor: Gerson Risso\*/**

**#include<stdio.h>**

**main(){**

**system(“color 4a”);**

**system(“title Primeiro Exemplo”);**

**printf(“Gerson Risso\n”);**

**system(“pause”);**

**}**

Equivale em Português estruturado

**principal(){**

**escreva(“Gerson Risso”);**

**}**

**Variável** é um espaço **alocado** na memória com **tipo de dado específico.** Associado a este espaço está o **identificador da variável** e o **seu endereço físico.**

Exemplo de declaração de variável: **float x=345.6;**

Representação de uma variável alocada na memória

Os identificadores podem ser nomeados como o programador desejar, desde que siga algumas regras básicas:

* Nunca use acentuação em nomes de variáveis;
* Nunca utilize caracteres especiais nos nomes das variáveis (A exceção é \_ (underline));
* Não inicie um identificador com número;
* Crie nomes coerentes e curtos se possível;

Exemplos de nomes de variáveis **INCORRETAS**

a) 12n;

b) nome de funcionário;

c) nome-do-filho;

Exemplos de nomes de variáveis **CORRETAS**

a) n12;

b) nomedofuncionario;

c) nomeDoFilho;

d) salario\_base;

**Observação:** Embora o exemplo b) não esteja incorreto, é inconveniente. Pois, dificulta a sua leitura.

Já os exemplos c) e d) são mais legíveis.

**Operador atribuição**

|  |  |  |
| --- | --- | --- |
| **Variável** | **Operador** | **Valor** |
| n | **=** | 23 |

**Segundo Exemplo**

Escreva um algoritmo em C que leia dois números reais, calcule a soma entre eles e exiba o resultado.

#include<stdio.h>

main(){

/\*Declaração

de variáveis\*/

float a,b,c;

printf(“Digite dois numeros\n”);

scanf(“%f%f”,&a,&b);

c=a+b;

printf(“Soma e %.2f\n”,c);

system(“pause”);

}

Exemplo 3: Escreva um algoritmo em C que leia dois números reais, calcule o produto entre eles e mostre o resultado na tela.

#include<stdio.h>

main(){

float a,b,c;

printf(“Digite dois numeros\n”);

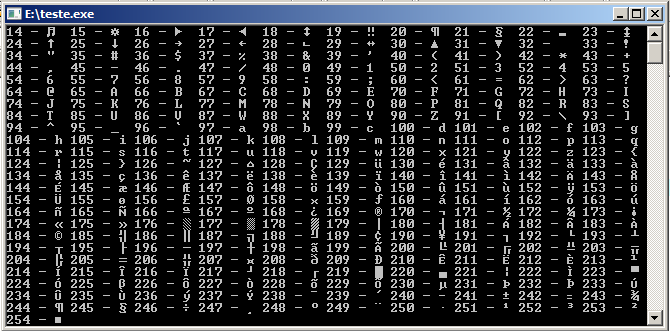
scanf(“%f%f”,&a,&b);

c=a\*b;

printf(“Resultado:%.2f\n”,c);

system(“pause”);

}



**Uma parte da tabela ASCII**

A Linguagem C foi criada por Dennis Ritchie e foi baseada na linguagem antecessora B, que, por sua vez, baseou-se na linguagem BCPL (Kernighan, B. W.:Ritchie, D. M.: C A Linguagem de programação padrão ANSI. Campus).

Podemos considerar que a linguagem C é de médio nível, podendo ser utilizada, portanto, em alto nível e baixo nível.

Após criamos o código-fonte, devemos compilá-lo, a fim de que o algoritmo seja executado. Dessa maneira, teremos dois arquivos, que consistem no código-fonte e no código compilado. Portanto, torna-se uma linguagem nativa da máquina.

A linguagem C possui símbolos específicos que representam os operadores aritméticos básicos, operadores relacionais e operadores lógicos.

A linguagem C é *case sensitive*, isto é, a linguagem faz distinção entre letras maiúsculas e minúsculas. Assim sendo, **A** e **a** são variáveis diferentes.

O IDE (Integrated Development Environment) que utilizaremos em nossas aulas é o Dev C++, disponível no link. Veja a versão mais adequada ao seu sistema operacional.

[**http://www.bloodshed.net/devcpp.html**](http://www.bloodshed.net/devcpp.html)

**Operadores**

* Aritméticos
* Relacionais
* Lógicos

**Operadores aritméticos**

A seguir mostramos os símbolos para os operadores aritméticos básicos.

|  |  |  |  |
| --- | --- | --- | --- |
| **Operador** | **Em C** | **Português estruturado** | **Exemplo** |
| Soma | + | + | r=a+b |
| Subtração | – | – | d=c–b |
| Multiplicação | \* | \* | m=a\*b |
| Divisão | / | / (real) div(inteiro) | h=a/b (1) |
| Módulo | % | mod | a=s%t (2) |

(1) O símbolo / em C, pode representar divisão inteira ou real, dependendo do tipo de dados das variáveis envolvidas.

Exemplo: c=a/b, sendo a=3, b=2 e c variáveis do tipo inteiro. Assim, c=1. A parte fracionária é descartada!

Mas considerando, as mesmas variáveis, do tipo real. Temos c=1.5.

(2) O símbolo % representa uma divisão inteira que retorna o resto. Sendo assim, 4%3 (diz-se: quatro mod três) é igual a um.

O algoritmo em português estruturado, a seguir, determina o resto da divisão entre dois números fornecidos pelo usuário.

principal(){

inteiro n1,n2,r;

escreva(“Digite dois números”);

leia(n1,n2);

r<-n1 mod n2;

escreva(“O resto é”,r);

}

**Exemplos de Algoritmos em C e português estruturado**

1) Escreva um algoritmo em:

* Português Estruturado;
* Linguagem C;

para ler dois números reais, calcular a soma e exibir o resultado.

**Solução**

Português Estruturado

principal(){

real a,b,c;

escreva(“Digite dois números reais”);

leia(a,b);

c<-a+b;

escreva(c);

}

**Tipos de dados em Linguagem C**

**char** – Variáveis declaradas com esse tipo de dado armazenam caracteres.Porém, uma variável do tipo **char**, armazena apenas um caractere por vez.

**int** – declaradas com esse tipo de dado armazenam números inteiros, positivos ou negativos.

**float** – Variáveis declaradas com esse tipo de dado armazenam números reais (positivos e negativos) de simples precisão.

**double** – Variáveis declaradas com esse tipo de dado armazenam números reais (positivos e negativos) de dupla precisão.

**Exemplo: Elabore um algoritmo em:**

* Português estruturado
* Fluxograma
* **Linguagem C**

Que leia o preço de um produto, calcule o desconto de 5% e exiba o preço com desconto.

#include<stdio.h>

main(){

float preco, precoDesc;

printf(“Digite o preco do produto\n”);

scanf(“%f”,&preco);

precoDesc=preco-preco\*0.05;

printf(“Preco com desconto R$%f ”,precoDesc);

system(“pause”);

}

**Tabela de correspondência entre o tipo e o parâmetro**

|  |  |
| --- | --- |
| Tipo primitivo | Parâmetro |
| char | %c |
| int | %d |
| float | %f |
| double | %lf |

Exercício

Atribua os valores adequados para as variáveis declaradas que são apresentadas a seguir.

a) float m=

b) int g=

c) double c=

d) int h=

e) char n=

f) char t=

g) float r=

**Comandos de entrada e saída de dados**

Os comandos de entrada e saída de dados, oferecem uma interface para a interação humana na execução do programa.

Em C existem várias funções de entrada e de saída, cada uma delas com suas peculiaridades. Porém, devemos destacar duas funções básicas:

**printf()** – Exibe no monitor (dispositivo padrão) um conjunto de caracteres, valores de variáveis ou os dois tipos de informações de maneira concatenada.

**scanf()** – lê informações inseridas, durante a execução do algoritmo, via teclado (dispositivo padrão).

As duas funções pertencem à biblioteca stdio.h. Portanto, é necessário inserir a diretiva de pré-processamento no algoritmo.

**A estrutura básica da linguagem C**

Qualquer algoritmo escrito em C, é composto por módulos e o principal é o **main()**. Com efeito, ele é imprescindível para a execução do algoritmo.

main(){

}

Escrito dessa maneira, não há como criar algoritmos interessantes, já que as funções e comandos estão contidos em bibliotecas. Sendo assim, a diretiva *#include<stdio.h>* será sempre útil para compilar e executar algoritmos, utilizando o IDE Dev C++.

#include<stdio.h>

main(){

}

Exemplo 3: Elabore um algoritmo em C que leia dois números inteiros, some-os e apresente o resultado na tela.

#include<stdio.h>

main(){

system(“title Exemplo3”);

system(“color 4b”);

int x,y,z;

printf(“Digite dois numeros\n”);

scanf(“%d%d”,&x,&y);

z=x+y;

printf(“Soma=%d\n”,z);

system(“pause”);

}

Exemplo 4: Escreva um algoritmo em C que leia a idade de uma pessoa, determine quantos dias ela viveu, aproximadamente e mostre o resultado na tela.

#include<stdio.h> //diretiva

main(){

int idade,quant;

printf(“Informe a sua idade: ”);

scanf(“%d”,&idade);

quant=idade\*365;

printf(“Voce ja viveu aproximadamente %d dias\n”,quant);

system(“pause”);

}

Fazer o exercício 1 e enviar para o E-mail [gerson-risso@uninove.br](mailto:gerson-risso@uol.com.br), um arquivo word com:

* Código-fonte em C;
* Português Estruturado;
* Fluxograma (Diagrama de blocos);

Atenção: no assunto escreva assim: **Uninove - Terça - Manhã**

Entregar até 12/09/2014

**Exercício 1**: Escreva um algoritmo em:

* Linguagem C;
* Português Estruturado;
* Fluxograma;

Para ler a temperatura em Fahrenheit, converter a temperatura para graus Celsius e exibir a temperatura convertida.

Dada a relação para conversão entre as temperaturas mencionadas.

**Exercício 2: Elabore um algoritmo em:**

* **Linguagem C;**
* **Português Estruturado;**
* **Fluxograma;**

**Para ler o peso e altura de uma pessoa, calcular o IMC (índice de massa corpórea) e exibir o resultado na tela.**

**Dada a relação matemática para cálculo do IMC:**

**Exercício 2: Elabore um algoritmo em:**

* **Linguagem C;**
* **Português Estruturado;**
* **Fluxograma;**

**Para ler a idade e nome de uma pessoa e exibir a quantidade de dias vividos e o nome da pessoa, na tela.**

**Operadores relacionais**

Os operadores relacionais são utilizados quando desejamos fazer comparações entre valores, sejam eles numéricos ou não.

|  |  |  |  |
| --- | --- | --- | --- |
| **Operador** | **Em C** | **Português estruturado** | **Exemplo** |
| Maior que | > | > | a>b |
| Menor que | < | < | a<b |
| Igual a | == | = | a==b |
| Diferente de | != | <> | a!=b |
| Maior que ou igual a | >= | >= | a>=b |
| Menor que ou igual a | <= | <= | a<=b |

A avaliação de uma dada comparação, utilizando um operador relacional, devolve sempre um valor lógico.

**Exercício:** Resolva as expressões a seguir.

a) 34%3

b) 3/2 (inteiros)

c) 5/2 (reais)

d) 5+3>3–2

e) 18%5>=5–2

f) 2\*3!=7

g) (12+7)%3<=1

**Operadores lógicos**

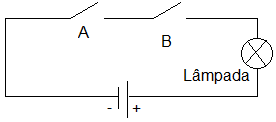
Fazem a associação de expressões que estabelecem uma relação de comparação entre valores. O resultado de tais expressões é sempre um valor lógico.

|  |  |  |
| --- | --- | --- |
| **Operador** | **Símbolo em C** | **Exemplo** |
| e | && | a+b>5 && c–d==e |
| ou | || | a+b>5 || c–d==e |
| não | ! | ! a==b |

**Tabela Verdade**

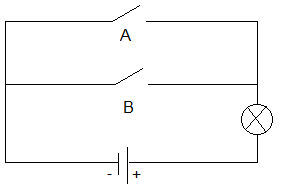
Representa todas as possibilidades existentes para a combinação de variáveis ou expressões utilizando-se um determinado operador lógico.

**Operador && (e)**



|  |  |  |
| --- | --- | --- |
| A | B | A && B |
| 0 | 0 | 0 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 1 | 1 |

**Operador || (ou)**



|  |  |  |
| --- | --- | --- |
| A | B | A || B |
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 1 |

**Operador ! (não)**

|  |  |
| --- | --- |
| A | !A |
| 0 | 1 |
| 1 | 0 |

**Exercício:** Resolva as expressões mistas apresentadas a seguir. Considere que as variáveis possuem os seguintes valores: a=3, b=4, c=8, x=3, y=2.

a) a>3 && c==8

b) a!=2 || b<=5

c) a==3 || b>=2 && c!=8

d) a==3 && !b<=4 && c==8

e) !(c==b) && a+b<=20 || a==b

f) x>y && c<=b

i) ! (c==b) && x+y<=20 || x>y

**Estruturas de decisão (seleção)**

São estruturas utilizadas para selecionar a instrução ou o bloco de instruções que será executado segundo uma ou mais condições pré-estabelecidas.

**A estrutura *if*.**

Forma geral

Para uma instrução

if(<condição>)

instrução1;

Para mais de uma instrução

if(<condição>){

instrução11;

instrução12;

}

**.**

**.**

**Forma geral**

**Para uma instrução**

**if(<condição>)**

**instrução1;**

**else**

**instrução2;**

**Para mais de uma instrução**

**if(condição){**

**instrução11;**

**instrução12;**

**}**

**else{**

**instrução21;**

**instrução22;**

**}**

**Exemplo 1:** Escreva um algoritmo em

* Linguagem C;
* Fluxograma;
* Português Estruturado;

que leia um número inteiro, **verifique** se ele é par e **mostre** uma mensagem informando o usuário.

Em C

#include<stdio.h>

main(){

int n;

printf(“Informe um numero\n”);

scanf(“%d”,&n);

if(n%2==0)

printf(“Par\n”);

system(“pause”);

}

Português Estruturado

principal(){

inteiro n,r;

escreva(“Informe um número”);

leia(n);

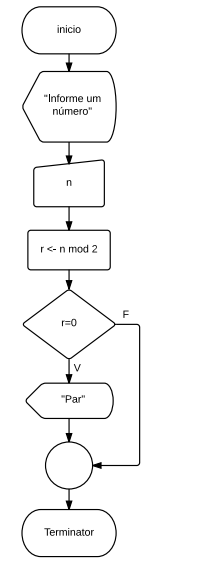
r<-n mod 2;

se (r=0)

escreva(“Par”);

}

Fluxograma



**Exemplo 2:** Escreva um algoritmo em:

* Linguagem C;
* Fluxograma;
* Português Estruturado;

que leia um número inteiro, verifique se é par ou ímpar e avise o usuário.

Em C

#include<stdio.h>

main(){

int n;

printf(“Informe um numero\n”);

scanf(“%d”,&n);

if( n%2==0)

printf(“Par\n”);

else

printf(“Impar\n”);

system(“pause”);

}

Em Português Estruturado

principal(){

inteiro n,r;

escreva(“Informe um número”);

leia(n);

r<- n mod 2;

se (r=0)

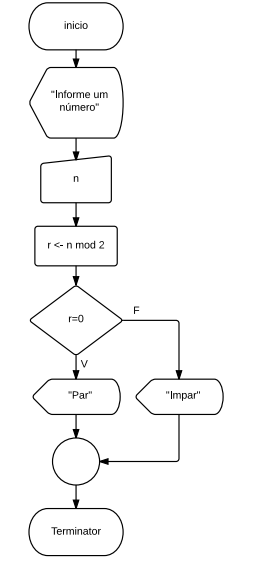
escreva(“Par”);

senão

escreva(“Impar”);

}

Fluxograma



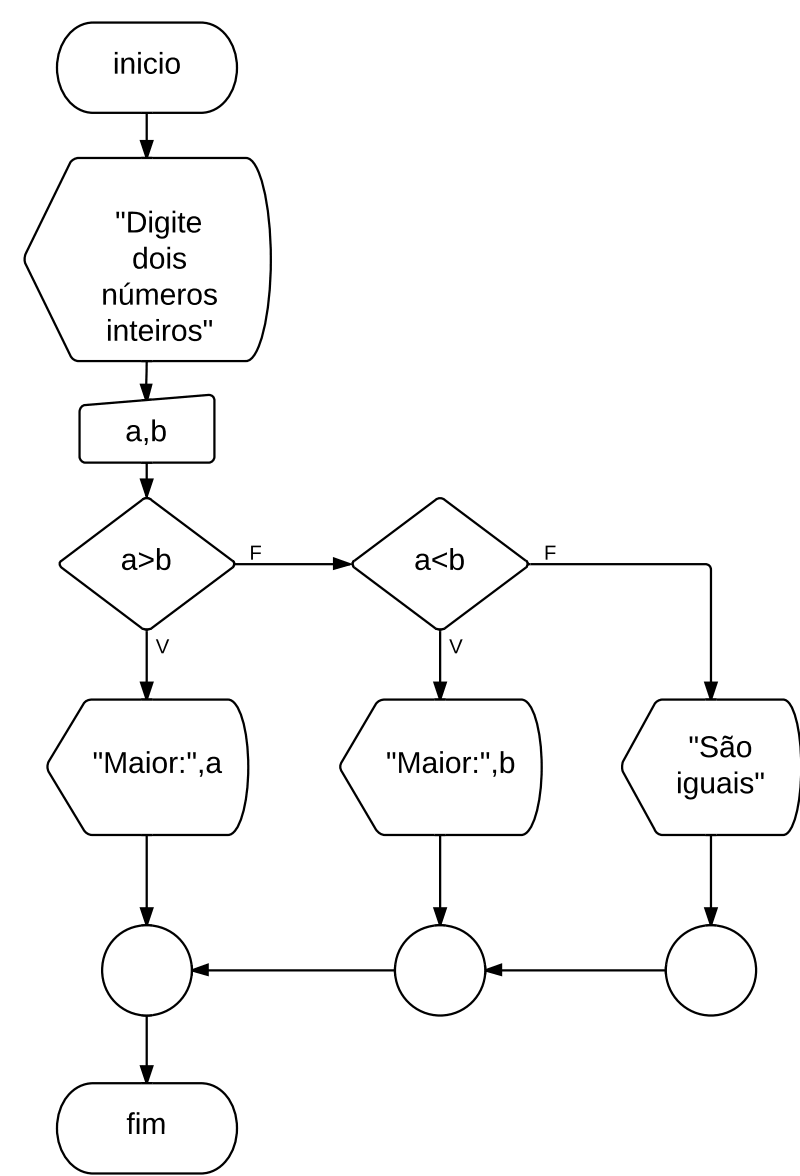
**Exercícios**

**1)** Um aluno realizou **três provas** de uma determinada disciplina e sabe-se que para ele ser **aprovado**, deverá atingir **média de no mínimo 6**. Escreva um algoritmo em C que **leia três notas de provas**, **calcule a média aritmética** simples, **verifique** se o aluno está ou não aprovado e **exiba** as informações na tela.

**2)** Escreva um algoritmo:

* em C
* fluxograma
* português estruturado

que leia dois números inteiros, verifique o maior entre eles e exiba-o na tela.



principal(){

inteiro a,b;

escreva(“Informe dois números inteiros”);

leia(a,b);

se (a>b)

escreva(“Maior:”,a);

senao se (a<b)

escreva(“Maior:”,b);

senao

escreva(“São iguais”);

}

**3)** Uma loja de departamentos promoveu uma semana de ofertas em seus produtos, conforme as condições:

**preço do produto até R$ 50,00 - desconto de 3%**

**preço do produto maior que R$50,00 - desconto 5%**

Elabore um algoritmo que **leia o preço** do produto, realize o **desconto** e **mostre** o o preço com desconto.

**Exemplo: O algoritmo a seguir lê o nome e a idade de uma pessoa e exibe essas informações na tela.**

**#include<stdio.h>**

**main(){**

**char nome[60];**

int idade;

printf(“Informe a idade\n”);

scanf(“%d”,&idade);

printf(“Informe o nome\n”);

**scanf(“%s”,&nome);**

printf(“Nome:%s \t Idade:%d\n”,nome,idade);

system(“pause”);

}

**4) Uma empresa por ocasião da comemoração do seu centenário, resolveu bonificar os seus funcionários, segundo a tabela:**

|  |  |
| --- | --- |
| **Tempo de serviço (anos)** | **Bônus (%)** |
| acima de 8 | 8 |
| entre 8 e 4 | 3 |
| abaixo de 4 | 1 |

O bônus é calculado segundo a regra:

Escreva um algoritmo em C que leia o tempo de serviço, o salário e o nome, calcule o bônus adequado e exiba o salário bonificado.

No algoritmo em C faça como no modelo:

/\*UNINOVE - SI - SEXTA - MANHÃ

Joãozinho da Esquina RA: 123456789

Mariazinha Sei Lá RA: 987654321\*/

**5)** Elabore um algoritmo em C que simule uma calculadora com as quatro operações aritméticas básicas: adição, subtração, multiplicação e divisão. O usuário deverá escolher, através de um menu na tela (veja a sugestão), a operação desejada e em seguida deverá digitar os números para realizar a operação escolhida.

**Sugestão de menu**

**[ + ] Adição**

**[ - ] Subtração**

**[ \* ] Multiplicação**

**[ / ] Divisão**

**Correção**

#include<stdio.h>

main(){

float n1,n2,r;

char opr;

printf(“[ + ] - Adicao\n[ - ] - Subtracao\n[ \* ] - Multiplicacao\n[ \ ] - Divisao\n”);

printf(“<< ”);

scanf(“%c”,&opr);

if(opr==’+’){

printf(“Digite dois numeros\n”);

scanf(“%f%f”,&n1,&n2);

r=n1+n2;

printf(“Resultado: %.4f\n”,r);

} else if(opr==’-’){

printf(“Digite dois numeros\n”);

scanf(“%f%f”,&n1,&n2);

r=n1-n2;

printf(“Resultado: %.4f\n”,r);

}else if(opr==’\*’){

printf(“Digite dois numeros\n”);

scanf(“%f%f”,&n1,&n2);

r=n1\*n2;

printf(“Resultado: %.4f\n”,r);

}else if(opr==’/’){

if(n2==0)

printf(“Divisao por zero!\n”);

else{

printf(“Digite dois numeros\n”);

scanf(“%f%f”,&n1,&n2);

r=n1/n2;

printf(“Resultado: %.4f\n”,r);

}

}else

printf(“Operacao invalida!\n”);

system(“pause”);

}

**Estruturas de múltiplas escolhas**

**A estrutura *switch...case* (escolha...caso)**

Essa estrutura é útil para situações onde há múltiplas escolhas, pois o código escrito com essa estrutura fica mais inteligível.

**switch(variável){**

**case expressão/constante: instruções;**

**break;**

**case expressão/constante: instruções;**

**break;**

**default: instruções;**

**}**

**Solução com switch...case**

#include<stdio.h>

main(){

float n1,n2,r;

char opr;

printf(“[ + ] - Adicao\n[ - ] - Subtracao\n[ \* ] - Multiplicacao\n[ \ ] - Divisao\n”);

printf(“<< ”);

scanf(“%c”,&opr);

**switch(opr){**

**case ‘+’:** printf(“Digite dois numeros\n”);

scanf(“%f%f”,&n1,&n2);

r=n1+n2;

printf(“Resultado: %.4f\n”,r);

break;//interrompe da estrutura switch...case

case ‘-’: printf(“Digite dois numeros\n”);

scanf(“%f%f”,&n1,&n2);

r=n1-n2;

printf(“Resultado: %.4f\n”,r);

break;

case ‘\*’: printf(“Digite dois numeros\n”);

scanf(“%f%f”,&n1,&n2);

r=n1\*n2;

printf(“Resultado: %.4f\n”,r);

break;

case ‘/’: printf(“Digite dois numeros\n”);

scanf(“%f%f”,&n1,&n2);

if(n2==0)

printf(“Divisao por zero!\n”);

else{

r=n1/n2;

printf(“Resultado: %.4f\n”,r);

break;

}

**default: printf(“Operacao invalida!\n”);**

**}**

**system(“pause”);**

**}**

**Orientações:**

* Enviar para o E-mail: **gerson-risso@uol.com.br**
* No E-mail coloque no assunto:

**UNINOVE - Ciência da Computação - SEXTA - Noturno**

* No algoritmo coloque como comentário:

/\*Joãozinho da Esquina RA: 123456789

Mariazinha Sei Lá RA: 987654321\*/

Exercício: Elabore um algoritmo em:

Linguagem C;

Português Estruturado;

Fluxograma;

Para ler um número, verificar se ele é positivo ou negativo e informar o usuário.

Solução

Em C

#include<stdio.h>

main(){

float n;

printf(“Digite um numero\n”);

scanf(“%f”,&n);

if(n>=0)

printf(“Positivo”);

else

printf(“Negativo”);

system(“pause”);

}

Exemplo 3: Elabore um algoritmo em C que **leia** **dois números reais**, **compare-os** e mostre o número de **menor valor,** na tela.

#include<stdio.h>

main(){

float x,y;

printf(“Informe dois numeros\n”);

scanf(“%f%f”,&x,&y);

if(y>x)

printf(“Menor: %.3f\n”,x);

else if(x==y)

printf(“Iguais\n”);

else

printf(“Menor: %.3f\n”,y);

system(“pause”);

}

**Resolvendo o exemplo 1 com a estrutura if...else**

**Exemplo 1:** Escreva um algoritmo em

* Linguagem C;
* Fluxograma;
* Português Estruturado;

que **leia** um **número inteiro,verifique** se ele é par ou ímpar, e **mostre** uma mensagem informando o usuário.

Em C

#include<stdio.h>

main(){

int n,r;

printf(“Informe um numero inteiro\n”);

scanf(“%d”,&n);

r=n%2;

**if(r==0)**

**printf(“Par\n”);**

**else**

**printf(“Impar\n”);**

system("pause");

}

Português Estruturado

principal(){

inteiro n,r;

escreva(“Informe um número”);

leia(n);

r<-n mod 2;

**se (r=0)**

**escreva(“Par”);**

**senão**

**escreva(“Impar”);**

}

**Exercícios**

Fazer os exercícios 2 e 3 e enviar para o E-mail [gerson-risso@uol.com.br](mailto:gerson-risso@uol.com.br), três anexos:

* Linguagem C (DEV C++);
* Português Estruturado (Diagrama de blocos);
* Fluxograma (Visio).

O arquivos em C e português estruturado devem ser documentados, como o modelo:

/\*Gerson Risso RA:12345

Data 20/08/2013\*/

Atenção: no assunto escreva assim: Uninove - Tinf - Segunda -manhã.

Faça os seus código com indentação.

1) Faça um algoritmo em:

* Linguagem C;
* Português Estruturado;
* Fluxograma;

que leia um número inteiro, verifique se ele é divisível ou não, por três e mostre uma mensagem informativa ao usuário.

2) Elabore um algoritmo em:

* Linguagem C;
* Português Estruturado;
* Fluxograma;

que leia três notas de provas, calcule a média aritmética simples e verifique se aluno está ou não aprovado, segundo a condição: m<5, aluno reprovado.

**A estrutura *if...else* aninhada**

**Exemplo 1:** Uma instituição de ensino estabeleceu que o critério de aprovação de seus alunos seguiria às condições:

* média >= 7 - Aprovado
* 4<=média<7- Exame pode ser feito assim: **média >= 4 e média<7**
* média < 4 - Reprovado

Escreva um algoritmo em:

Linguagem C;

Português estruturado;

Fluxograma;

**Para ler três notas de provas,** **determinar a média aritmética simples** **e exibir, na tela, a situação do aluno.**

Em C:

#include<stdio.h>

main(){

float n1,n2,n3,m;

printf(“Digite as tres notas”);

scanf(“%f%f%f”,&n1,&n2,&n3);

m=(n1+n2+n3)/3;

if(m>=7)

printf(“Aprovado com média %f\n”,m);

else if(m>=4 && m<7)

printf(“Exame com média %f\n”,m);

else

printf(“Reprovado com média %f\n”,m);

system(“pause”);

}

Português Estruturado

principal(){

real n1,n2,n3,m;

escreva(“Digite as três notas de provas”);

leia(n1,n2,n3);

m<-(n1+n2+n3)/3;

**se** (m>=7)

escreva(“Aprovado com média=”,m);

**senão se** (4<=m e m<7)

escreva(“Exame com média=”,m);

**senão**

escreva(“Reprovado com média=”,m);

}

**Exercícios**

1) Uma loja de varejo promoveu uma semana de descontos em todos os seus produtos, segundo às condições:

**preço acima de R$100,00 - Desconto de 15%**

**preço entre R$100,00 e R$60,00 - Desconto de 8%**

**preço menor que R$ 60,00 - Desconto de 5%**

Escreva um algoritmo em:

Linguagem C

Português Estruturado

Fluxogram

Que leia o preço do produto, calcule o desconto adequado e exiba o preço com desconto.

Dada a relação para o cálculo :

precoDesc= preco - preco\*desconto

Exemplo de desconto de 5%

desconto=5/100

**Correção do exercício**

Em C

2) Uma empresa resolveu premiar os seus funcionários, por ocasião da comemoração de aniversário da sua fundação. A regra estabelecida segue as condições:

* Funcionário com tempo de serviço menor que cinco anos - Bônus de 3%
* Funcionário com tempo de serviço entre cinco anos e dez anos - Bônus de 8%
* Funcionário com tempo de serviço maior que dez anos - Bônus de 12%

Escreva um algoritmo em:

* Linguagem C
* Português Estruturado
* Fluxograma

Que leia o tempo de serviço, o salário e o nome do funcionário, determine o bônus, calcule o salário com bônus e exiba o salário atualizado.

Dado o cálculo do bônus:

sb=s+s\*bonus

Em C

#include<stdio.h>

main(){

float s,sb,b;

int temp;

char nome[45];

printf(“Informe o seu nome\n”);

gets(nome);

printf(“Informe o seu tempo de serviço\n”);

scanf(“%d”,&temp);

printf(“Informe o seu salário\n”);

scanf(“%f”,&s);

if(temp<5)

b=0.03;

else if(temp>=5 && temp<=10)

b=0.08;

else

b=0.12;

sb=s+s\*b;

**printf(“%s o seu salario atual e R$%.2f\n”,nome, sb);**

system(“pause”);

}

Em Português Estruturado

principal(){

real s,sb,b;

inteiro temp;

caracter nome;

escreva(“Informe o seu nome”);

leia(nome);

escreva(“Informe o seu tempo de serviço”);

leia(temp);

escreva(“Informe o seu salário”);

leia(s);

se (temp<5)

b<-0.03;

senão se (temp>=5 e temp <=10)

b<-0.08

senão

b<-0.12;

sb<-s+s\*b;

escreva**(nome**,” o seu salário atual é ”,**sb**);

}

Exemplo 3: Vamos alterar o enunciado do exemplo 1, para as seguinte condições:

O aluno estará reprovado se a sua nota for menor que quatro.

Vejamos como fica o algoritmo, segundo essas condições.

#include<stdio.h>

main(){

float p1,p2,p3,m;

printf("Digite as tres notas\n");

scanf("%f%f%f",&p1,&p2,&p3);

m=(p1+p2+p3)/3;

if(m>=4 && m<5) {

printf("Aluno em PERIODO DE REFORCO\n");

printf("Media:%0.1f\n",m);

}

else{

if(m>=5){

printf("Aluno esta PROMOVIDO\n");

printf("Media:%.1f\n",m);

}

else{

printf("Aluno esta REPROVADO\n");

printf("Media:%.1f\n",m);

}

}

system("pause");

}

Neste algoritmo, utilizamos a combinação de estruturas if...else. Ou seja, utilizamos uma estrutura if...else aninhada, ou encadeada.

Outro exemplo de uso de uma estrutura if...else aninhada.

**Exemplo**

Suponha uma lista com apenas quatro objetos, onde deseja-se exibir o nome do produto, quando for digitado o código corresponde. Conforme a tabela a seguir:

|  |  |
| --- | --- |
| **Produto** | **Código** |
| Borracha | 110 |
| Caneta | 123 |
| Caderno | 23 |
| Lápis | 455 |

Escreva um algoritmo em:

* Linguagem C;
* Português Estruturado;
* Fluxograma;

Para ler o código de um produto e exibir na tela o produto correspondente, conforme a tabela.

Solução em C

O algoritmo, utilizando a estrutura if...else aninhada fica assim,

#include<stdio.h>

main(){

int codigo;

printf(“Informe codigo\n”);

scanf(“%d”,&codigo);  
 **if(codigo==110)**

printf(“Borracha\n”);

**else if(codigo==123)**

printf(“Caneta\n”);

**else if(codigo==23)**

printf(“Caderno\n”);

**else if(codigo==455)**

printf(“Lapis\n”);

**else**

printf(“Produto nao cadastrado\n”);

system(“pause”);

}

Solução em Português Estruturado

principal(){

inteiro cod;

escreva(“Informe o código do produto”);

leia(cod);

se(cod=110)

escreva(“Borracha”);

senão se(cod=123)

escreva(“Caneta”);

senão se(cod=23)

escreva(“Caderno”);

senão se(cod=455)

escreva(“Lápis”);

senão

escreva(“Produto não existe”);

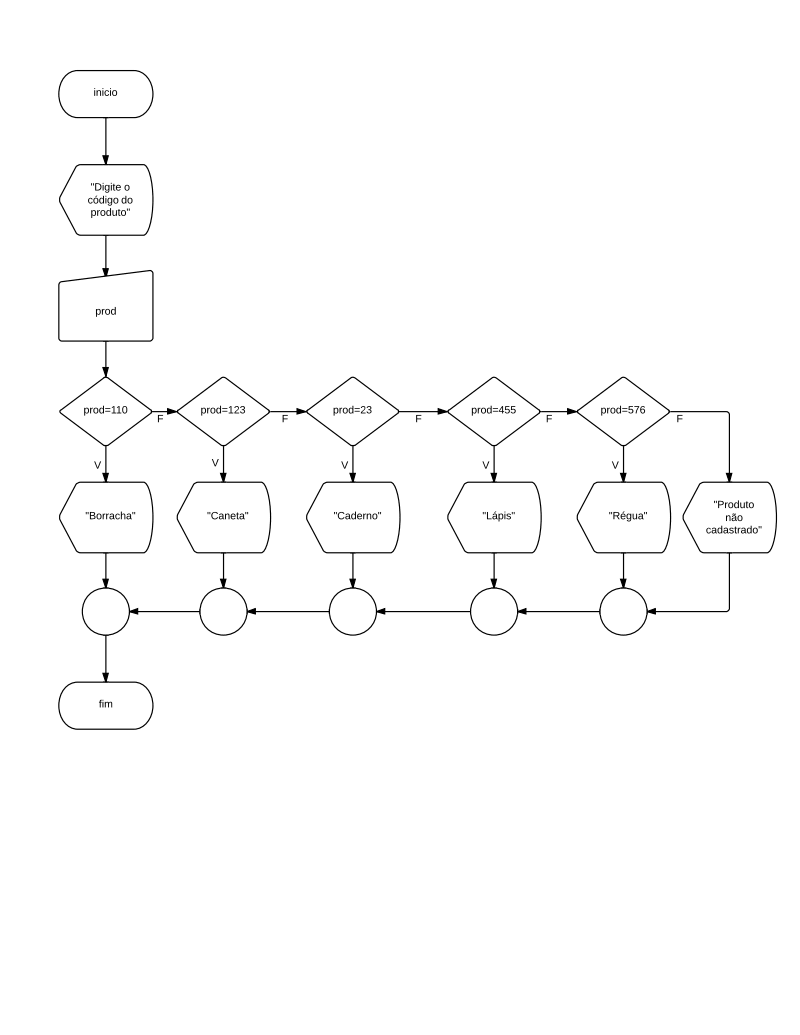
}

Não colocamos as chaves { } em if e else com apenas uma instrução. Pois, não é necessário e nesse caso, ajuda a diminuir a confusão com tantas chaves.

Imaginem como ficaria o algoritmo, com uma lista de apenas vinte produtos!

Existe uma estrutura, que para situações de múltiplas escolhas, é mais apropriado.

Fluxograma



**Estruturas de múltiplas escolhas**

**A estrutura *switch...case* (escolha...caso)**

Essa estrutura é útil para situações onde há múltiplas escolhas, pois o código escrito com essa estrutura fica mais inteligível.

**switch(variável){**

**case expressão/constante: instruções;**

**break;**

**case expressão/constante: instruções;**

**break;**

**default: instruções;**

**}**

Vejamos o algoritmo anterior, escrito com a estrutura switch...case.

**#include<stdio.h>**

**main(){**

**int codigo;**

**while(1){**

**system(“cls”);//apaga a tela**

**printf(“Informe codigo\n”);**

**scanf(“%d”,&codigo);**

**switch(codigo){**

**case 110: printf(“Borracha\n”);**

**break; //interrompe switch...case**

**case 123: printf(“Caneta\n”);**

**break;**

**case 23: printf(“Caderno\n”);**

**break;**

**case 455: printf(“Lapis\n”);**

**break;**

**case 0: exit(0); //interrompe a execução**

**default: printf(“Produto nao cadastro\n”);**

**}**

**system(“pause”);**

**}**

**}**

Português Estruturado

principal(){

inteiro cod;

escreva(“Digite o código do produto”);

leia(cod);

escolha(cod){

caso 23: escreva(“Caderno”);

interrompa;

caso 110: escreva(“Borracha”);

interrompa;

caso 123: escreva(“Caneta”);

interrompa;

caso 455: escreva(“Lápis”);

interrompa;

padrão: escreva(“Código incorreto!”)

}

}

Na forma geral da estrutura, não colocamos o comando *break* e nem o caso *default*, eles são opcionais. Porém, vale aqui, expor algumas ideias a respeito disso.

O comando break é utilizado para sair da estrutura, após a execução das instruções contidas no caso combinado com o argumento do comando switch. A sua ausência no algoritmo anterior, por exemplo, faria com que fossem executados todos os casos estabelecidos, após a combinação de um dos casos.

Com relação ao caso intitulado como default no algoritmo em questão, a sua função é apresentar uma mensagem se nenhum dos casos forem combinados. De um modo geral, então, o default será executado sempre que os casos estabelecidos não combinarem com o argumento do comando switch.

**Exercício: Elabore uma calculadora com as quatro operações aritméticas básicas. Monte um menu, através do printf, para exibir as informações. Sugestão para o menu:**

**[+] - Adicao**

**[-] - Subtracao**

**[/] - Divisao**

**[\*] - Multiplicacao**

**O usuário deve digitar dois números reais além de escolher a operação a ser realizada. Utilize a estrutura de múltipla escolha *switch..case*.**

**#include<stdio.h>**

**main(){**

**float x,y,z;**

**int opcao;**

**printf(“\t=== Menu ===\n”);**

**printf(“[1] - Adicao\n”);**

**printf(“[2] - Subtracao\n”);**

**printf(“[3] - Multiplicacao\n”);**

**printf(“[4] - Divisao\n”);**

**printf(“Escolha a operacao << ”);**

**scanf(“%d”,&opcao);**

**switch(opcao){**

**case 1: printf(“Digite dois numeros\n”);**

**scanf(“%f%f”,&x,&y);**

**z=x+y;**

**printf(“Resultado: %f\n”,z);**

**break;**

**case 2: printf(“Digite dois numeros\n”);**

**scanf(“%f%f”,&x,&y);**

**z=x-y;**

**printf(“Resultado: %f\n”,z);**

**break;**

**case 3: printf(“Digite dois numeros\n”);**

**scanf(“%f%f”,&x,&y);**

**z=x\*y;**

**printf(“Resultado: %f\n”,z);**

**break;**

**case 4: printf(“Digite dois numeros\n”);**

**scanf(“%f%f”,&x,&y);**

**if(y==0)**

**printf(“Divisao por zero!”);**

**else{**

**z=x/y;**

**printf(“Resultado: %f\n”,z);**

**}**

**break;**

**default: printf(“Esta operacao nao existe!”);**

**}**

**system(“pause”);**

**}**

**Estruturas de repetição em linguagem C**

**Laços condicionais**

**while**

Testa a condição no início e se a condição for verdadeira realiza as instruções, caso contrário vai para a próxima instrução fora do laço de repetição.

Forma geral

**while(condição){**

**instruções;**

**}**

do...while

Realiza as instruções na primeira passagem e testa a condição no fim, se a condição for verdadeira realiza as instruções novamente, caso contrário vai para a próxima instrução fora do laço de repetição.

**Forma geral**

**do{**

**instruções;**

**}while(condição);**

**Exemplos**

1) Escreva um programa em:

* Linguagem C;
* Fluxograma;
* Português Estruturado.

Que gere e exiba na tela os 100 primeiros números, começando por 0 (zero).

#include<stdio.h>

#define MAX 100

main(){

int a=0;

while(a<MAX){

printf(“%d\t”,a);

a=a+1;

}

system(“pause”);

}

1. Elabore um algoritmo em C que gere e exiba a tabuada do nove.
2. **Execute o algoritmo a seguir:**

**#include<stdio.h>**

**#include<time.h>**

**main(){**

**system("color 0A");**

**srand(time(NULL));**

**int i;**

**while(1){**

**i=rand()%127;**

**sleep(2);**

**printf("%c\t",i);**

**}**

**system("pause");**

**}**

**Exemplo 4:** Elabore um algoritmo em C que **leia dez números inteiros** e para cada um deles determine se é par ou ímpar.

**#include<stdio.h>**

**#define MAX 10**

**main(){**

**int n,i=1;**

**printf(“Informe %d numeros\n”,MAX);**

**do{**

**system(“cls”); //apaga a tela**

**scanf(“%d”,&n);**

**if(n%2==0)**

**printf(“Par\n”);**

**else**

**printf(“Impar\n”);**

**i++;**

**system(“pause”);**

**}while(i<=MAX);**

**}**

**Laço contado**

for

Realiza o laço de repetição, *n* vezes. Sendo *n* um número de iterações pré especificadas.

Forma geral

Para uma instrução

**for(valor inicial;critério de parada;controle)**

**instrução;**

Para mais de uma instrução

**for(valor inicial;critério de parada;controle){**

**instruções;**

**}**

Exemplo 1: Gerar e exibir a série: 2 4 6 8 ... 100

#include<stdio.h>

#define TAM 100

main(){

int a;

for(a=2;a<=TAM;a+=2)

printf(“%d\t”,a);

system(“pause”);

}

principal(){

inteiro a;

para(a<-2;a<=100;a=a+2)

escreva(a);

}

**Exemplos**

1) Escreva um programa em C que gere e exiba na tela os 100 primeiros números, começando por 0 (zero).

#include<stdio.h>

#define MAX 100

main(){

int a=0;

while(a<MAX){

printf(“%d\t”,a);

a=a+1;

}

system(“pause”);

}

Em português Estruturado

principal(){

inteiro a;

a<-0;

enquanto (a<100){

escreva(a);

a<-a+1;

}

}

**Formas abreviadas de expressões**

|  |  |
| --- | --- |
| Expressão | Forma Abreviada |
| n=n+1 | n+=1 ou n++ ou ++n |
| x=x-1 | x-=1 ou x-- ou --x |
| a=a+10 | a+=10 |
| b=b/5 | b/=5 |
| c=c\*8 | c\*=8 |

Exemplo 2: Elabore um algoritmo em C que gere e exiba a série numérica finita:

2 4 6 8 ... 300

Em C

#include<stdio.h>

#define MAX 300

main(){

int i=2;

while(i<MAX){

printf(“%d\n”,i);

i+=2;

}

system(“pause”);

}

Em português Estruturado

principal(){

inteiro a;

a<-2;

enquanto (a3100){

escreva(a);

a<-a+2;

}

}

**Exercícios**

1. **Escreva um algoritmo em:**

* **Linguagem C;**
* **Português Estruturado**
* **Fluxograma**

**para gerar e exibir na tela, as séries numéricas finitas:**

**a) 100 98 96 ... 0**

**b) 1 3 5 ... 311**

**c) -1 2 -3 4 -5 ... 200**

1. **Elabore um algoritmo que leia a tabuada desejada, gere e exiba os resultados na tela.**
2. Escreva um algoritmo em:

* Linguagem C;
* Português Estruturado
* Fluxograma

para ler trezentos números reais, calcular e exibir a média aritmética simples.

**Solução:**

**1) a)**

Em C

#include<stdio.h>

main(){

int n=100;

while(n>=0){

**printf(“%d\t”, n);**

**n-=2;//n=n-2;**

}

system(“pause”);

}

Em português estruturado

principal(){

inteiro n;

n<-100;

enquanto(n>=0){

escreva(n);

n<-n-2;

}

}

**1) b)**

Em C

#include<stdio.h>

#define MAX 311

main(){

int a=1;

while(a<=MAX){

printf(“%d\n”,a);

a+=2; //a=a+2;

}

system(“pause”);

}

Em português estruturado

principal(){

inteiro a;

a<-1;

enquanto(a<=311){

escreva(a);

a<-a+2;

}

}

1) c)

#include<stdio.h>

#define T 200

main(){

int i=1,r;

while(i<=T){

r=i%2;

if(r==1)

printf(“-%d\t”, i);

else

printf(“%d\t”,i);

i++;

}

system(“pause”);

}

Em português estruturado

principal(){

inteiro i,r;

i<-1;

enquanto(i<=200){

r<-i mod 2;

se(r=1)

escreva(-i);

senao

escreva(i);

i<-i+1;

}

}

A estrutura de repetição **do...while**

Forma geral

**do{**

**instruções;**

**}while(condição);**

**Exemplo 1:** Elabore um algoritmo em C que gere e exiba a série numérica finita:

2 4 6 8 ... 300

#include<stdio.h>

#define MAX 300

main(){

int i=2;

**do{**

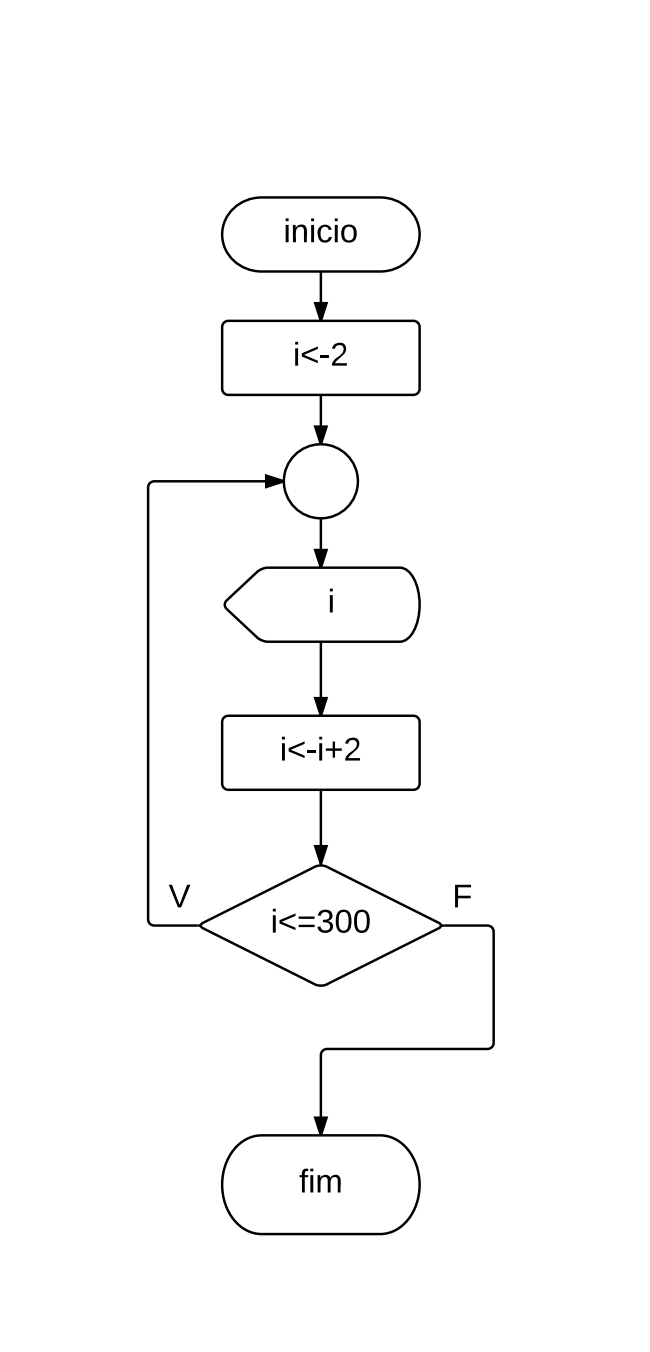
printf(“%d\t”,i);

i+=2;

}while(i<=MAX);

system(“pause”);

}



Português estruturado

principal(){

inteiro i;

i<-2

faça{

escreva(i);

i<-i+2;

}enquanto(i<=300);

}

**Exercícios**

1. **Escreva um algoritmo em C que gere as séries:**
2. **1 2 3 4 5 ... 100**
3. **100 99 98 97 ... 1**
4. **1 1 2 2 3 3 ... 100 100**
5. **1 3 5 7 ... até o termo que desejar**
6. **1 1 2 3 5 8 13 21 ... até o termo que desejar**
7. **Elabore um algoritmo em C que leia o valor inicial da série e gere os dez números múltiplos do valor digitado.**
8. **Elabore um algoritmo em C que leia mil números reais, determine a média aritmética de todos os valores digitados e exiba o resultado na tela.**
9. **Uma turma de sessenta alunos realizou um prova e o professor da disciplina deseja saber a média de notas da turma. Escreva um programa que leia todas as notas, calcule a média e exiba-a na tela.**
10. **Uma empresa do setor elétrico mediu, no período de trinta dias, o consumo de energia elétrica de uma cidade. Ela deseja saber o consumo médio, o dia de maior consumo, o dia de menor consumo.**

**Estrutura de repetição com laço contado**

**Estrutura for**

**Forma geral**

Para uma instrução

**for(valor\_inicial;valor\_final;incremento/decremento)**

**instrução;**

Para mais de uma instrução usa-se, obrigatoriamente as chaves

**for(valor\_inicial;valor\_final;incremento/decremento){**

**instrução1;**

**instrução2;**

**instruçãon;**

**}**

**Exemplo 1:** Elabore um algoritmo em C que gere e exiba a série numérica finita:

2 4 6 8 ... 300

#include<stdio.h>

#define TAM 300

main(){

int i;

for(i=2;i<=TAM;i+=2)

printf(“%d\t”,i);

system(“pause”);

}

Português Estruturado

principal(){

inteiro i;

para (i2;300;2)

escreva(i);

}

**Exemplo 2:** Elabore um algoritmo em C que **leia dez números inteiros** e para cada um deles determine se é par ou ímpar.

#include<stdio.h>

#define TAM 10

main(){

int n,i;

**for(i=1;i<=TAM;i++){**

printf(“Digte %d numeros\n”,TAM);

scanf(“%d”,&n);

if(n%2==0)

printf(“Par\n”);

else

printf(“Impar\n”);

system(“pause”);

system(“cls”);//apaga a tela

**}**

}

**Exercícios**

**Observação:** Utilize a estrutura **do...while** pelo menos no item a do primeiro exercício.

1) Escreva um algoritmo em:

* Linguagem C;
* Português Estruturado
* Fluxograma

para gerar e exibir na tela, as séries numéricas finitas:

a) 100 98 96 ... 0

b) 1 3 5 ... 311

c) -1 2 -3 4 -5 ... 200

2) Escreva um algoritmo em:

* Linguagem C;
* Português Estruturado
* Fluxograma

para ler trezentos números reais, calcular e exibir a média aritmética simples.

3) Escreva um algoritmo em:

* Linguagem C;
* Português Estruturado
* Fluxograma

Para gerar e exibir na tela, as séries numéricas:

a) 1 1 2 2 3 3 ...20 20

b) 1 2 2 3 3 3 ... até o termo de valor igual a 20, que será exibido vinte vezes na tela.

c) 1 1 2 3 5 8 13 ... até trigésimo termo da série.

4) Uma turma de cinquenta alunos realizou a AV1 e o professor necessita calcular a média aritmética da turma.

Escreva um algoritmo em:

* Linguagem C;
* Português Estruturado
* Fluxograma

que leia as cinquenta notas, calcule a média da turma e exiba o resultado da média na tela.

5) Escreva um algoritmo em:

* Linguagem C;
* Português Estruturado
* Fluxograma

que o usuário insira a tabuada desejada e o algoritmo mostre todos os resultados da tabuada para o valor solicitado.

Exercícios Propostos

1. Escreva um programa em C que leia vinte códigos de produtos, conforme a tabela e exiba o nome do produto correspondente.

|  |  |
| --- | --- |
| **Produto** | **Código** |
| Borracha | 110 |
| Caneta | 123 |
| Caderno | 23 |
| Lápis | 455 |

1. Ler dez números reais, verificar qual o menor valor e exibi-lo na tela (não utilize vetor).
2. Gerar a série –1 2 –3 4 ... até o trigésimo termo.
3. Uma cidade A possui 25.000 habitantes e cresce a uma taxa de 0.5% ao ano. Uma cidade B vizinha à primeira possui 21.000 habitantes e cresce a uma taxa de 0.8%. Em quanto tempo a população da cidade B superará a população da cidade A? Faça um algoritmo em C que determine esse tempo.
4. Considere que digitou-se um número inteiro de dois dígitos. Escreva um algoritmo em C que leia esse número e mostre-o na tela, com os dígitos invertidos.
5. Escreva um algoritmo em C que leia dez números inteiros terminados com 9, caso contrário o algoritmo deve solicitar um novo valor ao usuário.
6. Faça uma calculadora aritmética com as quatro operações básicas. Possibilite ao usuário, digitar dois números quaisquer e a operação desejada. Utilize a estrutura switch..case.
7. Elaborar um algoritmo em C para calcular o produto entre os números inteiros 543 e 28. Exibir o resultado ao final.
8. Criar um algoritmo em C que calcule e exiba a média aritmética entre os números inteiros 46, 77 e 59.
9. Elaborar um algoritmo em C para ler um número inteiro, calcular seu sucessor e antecessor e mostrar ambos ao final.
10. Criar um algoritmo em C para ler um número real e exibir a terça parte deste número.
11. Desenvolver um algoritmo em C para calcular e exibir a nota final de um aluno. Leia a nota do primeiro semestre e a nota do segundo semestre, e considere que a primeira tem peso 40% e a segunda tem peso 60%. Utilizar a seguinte fórmula:

Nota Final = Nota1 \* 0,40 + Nota2 \* 0,60

1. Supondo que A, B e C são variáveis inteiras com valores iguais a 5, 10 e -8 respectivamente, e uma variável real D com valor igual a 1,5 , quais os resultados das expressões aritméticas a seguir ?

a. 2 \* A % 3 – C

b. ( (20 / 3) / 3 ) + ( 2 \* B ) / 2

c. (30 % 4 \* D) \* -1

1. Criar um algoritmo em C para ler uma temperatura em graus Celsius e apresentá-la convertida em graus Fahrenheit. A fórmula de conversão é:

F = ( 9 \* C + 160) / 5, sendo F a temperatura em Fahrenheit e C a temperatura em Celsius.

1. Elaborar um algoritmo em C que, dado o número de motos e o número de carros em um estacionamento, calcule e mostre os seguintes resultados:

a. Quantidade de pneus de motos

b. Quantidade de pneus de carros

c. Quantidade total de pneus

Quantidade de veículos estacionados (considerar motos e carros)

1. Elaborar um algoritmo em C que leia 2 variáveis inteiras e troque o conteúdo entre elas, sem utilizar uma terceira variável.
2. Criar um algoritmo em C que leia a quantidade de quilômetros percorridos, o tempo decorrido e quantos litros de combustível foram gastos em uma viagem. Calcule e informe a média de consumo do veículo e velocidade média.
3. Desenvolver algoritmo em C para ler quatro números (N1, N2, N3 e N4) e informar separadamente:

* A soma de N1, N2, N3 e N4;
* O resto da divisão de N1 por N3;
* A divisão de N4 por N1;
* A subtração de N4 por N1.

1. Criar um algoritmo em C para ler dois valores inteiros A e B e, se os valores forem iguais efetue a soma de A e B, caso contrário, multiplique A por B. Atribuir o resultado do cálculo para uma variável C e exibir o resultado.
2. Desenvolver um algoritmo em C para ler o nome e a idade de um cidadão brasileiro e informar seu nome e sua condição civil: maior de idade (a partir de 18 anos) ou menor de idade.
3. **Elaborar um algoritmo em C para ler o nome e 3 notas de um aluno. Informar seu nome e sua situação de acordo com a média obtida:**

* **“APROVADO” com média >=7**
* **“EXAME” com média < 7 e média >= 4**
* **“REPROVADO” com média < 4**

1. Desenvolver um algoritmo em C para ler a idade de um eleitor e informar se seu voto é obrigatório (idade de 18 até 59), opcional (idade de 16 até 18 e de 60 em diante) ou proibido (menor que 16).
2. **Criar um algoritmo em C para ler o nome e o código que determina o estado civil de um cliente e escrever o nome e a descrição conforme abaixo:**

* **“SOLTEIRO” à se código igual a “S”**
* **“CASADO” à se código igual a “C”**
* **“DIVORCIADO” à se código igual a “D”**
* **“VIUVO” à se código igual a “V”**

1. Elaborar um algoritmo em C para ler três números (A, B e C), os possíveis lados de um triângulo e exibir a classificação segundo os lados. Lembrando que:

* um triângulo é uma forma geométrica em que cada lado é menor que a soma dos outros dois lados;
* triângulo eqüilátero: todos os lados são iguais;
* triângulo isósceles: dois lados iguais e um lado diferente;
* triângulo escaleno: todos os lados são diferentes.

1. Criar um algoritmo em C que leia um número inteiro entre 1 e 12 e escreva o mês correspondente. Se for digitado um número fora deste intervalo, exibir uma mensagem informando que não há mês correspondente a este número.
2. Desenvolver um C para ler a sigla de um estado e escrever a capital correspondente, somente para os estados da região Sudeste. Caso seja digitada uma sigla inválida, exibir mensagem de erro.

Escreva um algoritmo em C que leia dez números inteiros, some-os e mostre o resultado na tela.

1. Escreva um algoritmo em C que leia vinte números reais, calcule a média e mostre o resultado na tela.
2. Elabore um algoritmo em C que leia vinte números inteiros e divisíveis por três, some-os e exiba o resultado.
3. Elabore um algoritmo em C que leia nomes, enquanto o usuário quiser, conte a quantidade de nomes e mostre o resultado na tela.
4. Elabore um algoritmo em C que leia números inteiros e encerre a leitura quando o usuário digitar um número terminado com nove. Calcule a média dos números digitados e mostre o resultado na tela.
5. Uma cidade A possui 25.000 habitantes e cresce a uma taxa de 0.5% ao ano. Uma cidade B vizinha à primeira possui 21.000 habitantes e cresce a uma taxa de 0.8%. Em quanto tempo a população da cidade B superará a população da cidade A? Faça um algoritmo em C que determine esse tempo.
6. Escreva um algoritmo em C que leia dez números reais e mostre, o maior valor digitado.
7. Escreva um algoritmo em C que leia um número inteiro e calcule o seu fatorial.
8. Escreva um programa em C para cada série:
9. 1 3 6 10 ...
10. 1 3 6 10 15 …
11. Escreva um algoritmo em C que determine se um número é primo ou não.
12. Escreva um programa em C que leia um número inteiro e determine o seu fatorial.
13. Foi realizada um eleição com dois candidatos:

* Mariazinha
* Joãozinho

Escreva um programa em C que simule a eleição, considerando que há a possibilidade do eleitor votar em branco ou nulo. A quantidade de eleitores não é definida, assim o usuário do sistema é que determinará quando o programa deve ser encerrado. O programa deve exibir os votos em cada candidato, votos em branco e votos nulos, além da quantidade de eleitores participantes.

algoritmo “Exemplo”

var

a: real

inicio(){

escreva(“Digite um número”)

leia(a)

escreva(“Você digitou ”, a)

}

#include<stdio.h>

#include<stdlib.h>

main(){

float a;

printf(“Digite um numero”);

scanf(“%f”,&a);

printf(“Voce digitou ”, a);

system(“pause”);

}

Todos os algoritmos têm três aspectos fundamentais. São eles: entrada de dados, processo e saída. Com efeito, para o primeiro algoritmo aqui mencionado (procedimento da troca de pneu furado) temos que a entrada é o pneu furado e as ferramentas utilizadas, o processo é a troca, efetivamente, do pneu furado e a saída é o carro em condições de uso.

Simbologia para o fluxograma

Símbolo

Significado

Explicação

Terminação

Utilizado para determinar o início e o fim de um algoritmo.

Processo

Representa a execução de operações como atribuição, soma e demais operações aritméticas.

Orientação do fluxo

Utilizado para indicar a direção e sentido do fluxo no algoritmo.

Entrada de dados

É um comando de entrada de dados e o dispositivo padrão é o teclado

Saída de dados

É um comando de saída de dados e dispositivo padrão é o teclado.

Decisão

Utilizado em processos de decisão.

Conector de página

Permite informar a origem do fluxo.

Conector de fluxo

Utilizado para realizar desvio ou interligação de fluxo.

Preparação

Utilizado para representar um processamento predefinido.

A estrutura básica de um pseudocódigo

Utilizaremos a seguinte estrutura básica

inicio(){

//Corpo do algoritmo

}

Os comandos básicos para este tipo de algoritmo são:

escreva() – Comando de saída de dados. Este comando pode exibir uma cadeia de caracteres e/ou valores armazenados em variáveis e o dispositivo padrão é o monitor de vídeo.

leia() – Comando de entrada de dados. É utilizado para inserir dados no algoritmo, em tempo de execução. O dispositivo padrão é o teclado.

Variáveis

As variáveis são espaços alocados (reservados) na memória e armazenam dados de tipos especificados previamente, no momento da declaração dessas variáveis.

Exemplos:

n,t:real

b:inteiro

Considerando estas declarações, podemos afirmar que existem três blocos alocados na memória. Dois estão preparados para receber valores do tipo real (n e t). E outro está preparado para receber valores do tipo inteiro.

O nome de variável (ou identificador) é livre, você poderá dar a esse bloco alocado, o nome que desejar. Porém, devemos seguir algumas regras básicas.

Regras para a criação de identificadores de variáveis

Nunca iniciar o nome de uma variável, com números ou símbolos especiais como @, # e? entre outros.

Maus exemplos: #re, 23n, 12.

Não acentue nomes de varáveis e não utilize espaços entre os caracteres que compõem o identificador.

Maus exemplos: funcionário, registro do funcionário, salário.

Se caso desejar um nome de variável compostos por duas ou mais palavras, pode-se usar esses recursos:

a) registro\_do\_funcionario (Não utilizar hífen)

b) registroDoFuncionario

c) registrodofuncionario

Embora não ocorra erro, a última sugestão não é apreciada, pois causa confusão na leitura. A segunda sugestão é desejável.

Não utilize palavras reservadas.

Maus exemplos: inteiro, inicio,var.

Utilize nomes coerentes e sucintos.

Constantes

Se em um algoritmo, um bloco alocado na memória, armazenar durante toda a execução, o mesmo valor, podemos denominá-lo constante. Ou seja, todas as regras válidas para as variáveis, aplicam-se igualmente às constantes.

Operadores aritméticos

A seguir mostramos os símbolos para os operadores aritméticos básicos.

Operador Símbolo Exemplo Soma + ra+b Subtração – dc–b Multiplicação \* ma\*b Divisão (1) / div ha/b ou h a div b Módulo (2) mod As mod t

(1) O símbolo / representa uma divisão real e o símbolo div representa uma divisão inteira.

Exemplo: ca div b, sendo a3, b2 e c variáveis do tipo inteiro. Assim, c 1. A parte fracionária é descartada!

Mas considerando, as mesmas variáveis, do tipo real. Temos ca div b e, portanto, c1.5.

(2) O símbolo mod representa uma divisão inteira que retorna o resto. Sendo assim, 4 mod 3 (diz-se: quatro mod três) é igual a um.

Exemplo: Desejamos soma dois números reais e armazenar o resultado em uma terceira variável. Para isto vamos escrever três tipos distintos de algoritmos.

Estruturas de Seleção (Decisão)

Utilizadas nas tomadas de decisão em programação. Em C, a forma geral de uma estrutura de decisão simples é

**Para uma instrução**

if(condição)

instrução;

**Para mais de uma instrução**

if(condição){

instruções;

}

A estrutura if...else tem a seguinte forma geral:

**Para uma instrução**

if(condição)

instruçãoA;

else

instruçãoB;

**Para mais de uma instrução**

if(condição){

instruçõesA;

}

else{

instruçõesB;

}

Exemplo 1: Escreva um programa em C que leia um número inteiro e determine se ele é par ou ímpar.

Solução:

#include<stdio.h>

main(){

int n,r;

printf(“Digite um número\n”);

scanf(“%d”,&n);

if(n%2==0)

printf(“Par\n”);

else

printf(“Impar\n”);

}

Exemplo 2: Escreva um programa em C que leia dois números reais e determine se eles são iguais ou diferentes.

#include<stdio.h>

main(){

float a,b;

printf(“Digite dois números\n”);

scanf(“%f%f”,&a,&b);

if(a==b)

printf(“Iguais\n”);

else

printf(“Diferentes\n”);

system(“pause”);

}

**Exercícios**

1) Escreva um algoritmo em C que dois números reais e exiba o maior entre eles.

2) Uma loja realizou uma semana de descontos em todos seus produtos. A regra utilizada foi:

produtos com preço:

menor que R$50,00, teve desconto de 5%

entre R$50,00 e R$100,00, teve desconto de 10%

maior que R$100,00, teve desconto de 25%

O exemplo 1, em fluxograma, é mostrado na Figura 1.

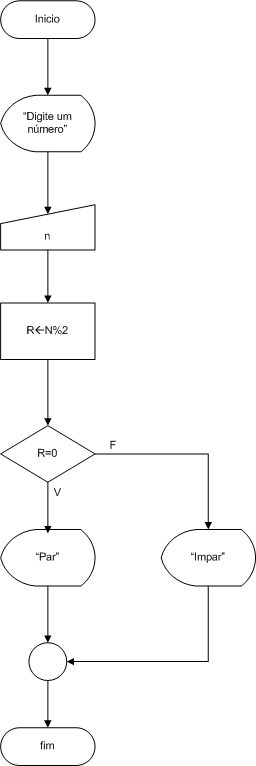


Figura 1 - Fluxograma do exemplo 1

Estruturas de repetição

**Laços condicionais**

while

Testa a condição no início e se a condição for verdadeira realiza as instruções, caso contrário vai para a próxima instrução fora do laço de repetição.

Forma geral

**while(condição){**

**instruções;**

**}**

do...while

Realiza as instruções na primeira passagem e testa a condição no fim, se a condição for verdadeira realiza as instruções novamente, caso contrário vai para a próxima instrução fora do laço de repetição.

Forma geral

**do{**

**instruções;**

**}while(condição);**

**Laço contado**

for

Realiza o laço de repetição, *n* vezes. Sendo *n* um número de iterações pré especificadas.

Forma geral

Para uma instrução

for(valor inicial;critério de parada;controle)

instrução;

Para mais de uma instrução

for(valor inicial;critério de parada;controle){

instruções;

}

**Estruturas de Dados Homogênea**

**Matrizes unidimensionais (vetores, *array*)**

Os vetores são estruturas de dados homogêneas que armazenam vários valores do mesmo tipo.

Exemplos de declaração de vetores:

**a) int v[10]; Primeiro índice:0 Último índice:9**

**b) float b[20];**  **Primeiro índice:0 Último índice:19**

c) **char nome[60];** **Primeiro índice:0 Último índice:59**

d) **double a[300];** **Primeiro índice:0 Último índice:299**

No exemplo *a* podemos representar o vetor pela tabela:

|  |  |
| --- | --- |
| **índice** | **Valor** |
| **0** | **3** |
| **1** | **454** |
| **2** | **5** |
| **3** | **29** |
| **4** | **9** |
| **5** | **88** |
| **6** | **0** |
| **7** | **-4** |
| **8** | **3** |
| **9** | **-10** |

Quando declaramos um vetor, alocamos na memória vários blocos com contiguidade (vizinhança) física e referenciados pelo nome. E cada bloco é associado a um índice, zero para o primeiro e, portanto, o último índice fica tamanho-1.

**Atribuição direta (Atribuição em tempo de compilação)**

**Exemplo:**

**int v[ ]={2,3,4,5,8,20};**

**Atribuição em tempo de execução**

Exemplos:

**scanf(“%f”,&v[4]);**

**for(i=0;i<30;i++)**

**v[i]=i;**

**Exibindo um conteúdo do vetor**

**Exemplos:**

**printf(“%d\n”,v[0]);**

**printf(“%c\n”,a[2]);**

**Exemplo 1:** Considerando a tabela a seguir, faça o que se pede:

Nome do vetor: **v**

|  |  |
| --- | --- |
| índice | Valor |
| 0 | 2 |
| 1 | 3 |
| **2** | **4** |
| 3 | 1 |
| 4 | 9 |
| 5 | 0 |
| 6 | 7 |
| 7 | 8 |
| 8 | 6 |
| 9 | 5 |

a) Declare um vetor correspondente à tabela.

**int v[10];**

**10];**

b) Qual é o valor armazenado em v[3], v[6] e v[v[0]]?

**Em v[3] armazena 1**

**Em v[6] armazena 7**

**Em v[v[0]] armazena 4**

c) Qual é o valor da operação v[v[2]]+v[v[1]]?

**Resultado: 10**

d) Escreva um comando para exibir o terceiro valor do vetor v.

**printf(“%d”,v[2])**

e) Escreva um comando para armazenar, via teclado, um valor inteiro no índice 5.

**scanf(“%d”,&v[5]);**

f) **Escreva um algoritmo em C que leia um vetor de cem números reais e exiba todos os valores armazenados neste vetor.**

#include<stdio.h>

#define MAX 100

main(){

float a[MAX];

int i;

printf(“Digite %d numeros\n”,MAX);

**for(i=0;i<MAX;i++)**

**scanf(“%f”,&a[i]);**

for(i=0;i<MAX;i++)

printf(“%.2f\n”,a[i]);

system(“pause”);

}

Português Estruturado

inicio()

real a[100];

inteiro i;

escreva(“Digite 100 números”);

para (i0;99;1)

leia(a[i]);

para (i0;99;1)

escreva(a[i]);

}

**Exemplo 2:** Escreva um algoritmo em:

* Linguagem C;
* Português Estruturado;
* Fluxograma;

que leia um vetor de vinte números inteiros. Exiba-os após o armazenamento.

Exercícios

1) Escreva um algoritmo em linguagem C que leia um vetor de cem números reais. Exiba os valores maiores que 10.

2) Elabore um algoritmo em C que leia um vetor de mil números reais, calcule a média dos números armazenados e exiba o resultado na tela.

3) Elabore um programa em linguagem C que armazene em um vetor de números inteiros e de tamanho igual a cem, valores correspondentes ao sucessor ímpar de cada índice deste vetor. Após o armazenamento dos valores, exiba o conteúdo do vetor na tela.

4) Escreva um algoritmo em C que gere e armazene em um vetor de quarenta números inteiros, a série numérica (Série de Fibonacci): 1 1 2 3 5 8 13 21...

5) Em uma região do Brasil foi avaliado o regime de ventos em um período de quatro meses (aproximadamente 120 dias). A fim de verificar a viabilidade de instalar turbinas eólicas para geração de energia elétrica.

Escreva um programa em linguagem C que leia, dia a dia, a velocidade dos ventos (em Km/h), determine a média das velocidades de vento, a quantidade de dias acima e abaixo da média, o maior e o menor valor de velocidade registrado.

Exiba na tela, todos as informações requisitadas, além de todos os registros de velocidade de vento.