SDAC

Linguagem C

# História da linguagem C

## História

## A linguagem C foi projectada em 1972 nos laboratórios Bell por Brian W. Kernighan para o sistema operativo UNIX, desenvolvido em 1967 e reescrito em 1973 utilizando a linguagem C, por Dennis M. Ritchie.

## Principais características da linguagem C

* Alto grau de portabilidade, que permite que os seus programas fonte sejam transportados entre máquinas sem maiores problemas
* É de uso geral, sendo eficiente tanto em programas de utilitários, como para sistemas operativos, processadores de texto, banco de dados e sistemas aplicativos em geral.
* Gera um código executável, compacto e rápido em relação à maioria das outras linguagens compiladas, tornando a sua aplicação eficiente para microcomputadores de memória reduzida.
* Possui uma sintaxe de poucos comandos e um grande numero de operadores aritméticos e lógicos.
* É uma linguagem estruturada e modular, o que permite a simplificação do uso das mais modernas técnicas de programação.
* Permite estruturas de dados compostos na forma de registos e campos.
* A linguagem C é constituída por funções sendo o próprio programa principal uma função (main( )).

# Algoritmos

## Definição

Podemos dizer que é uma forma ou fórmula para resolução de determinado problema, mediante o estabelecimento de determinadas regras ou procedimentos.

Em Informática, a formulação de um algoritmo não é mais do que a descrição, de forma ordenada, com clareza e rigor, das operações que se pretende realizar em computador para resolver um problema.

O desenvolvimento de um programa informático necessita sempre da formulação de um algoritmo.

Problema 🡺 Algoritmo 🡺 Programa

Identificação:

* dados
* operações
* resultados

Utitilização:

* C
* Pascal
* C++
* Basic
* Cobol
* Delphi
* etc

O algoritmo não deve ser escrito com palavras específicas da linguagem de programação escolhida, pelo contrário, um bom algoritmo deve ser escrito na nossa linguagem natural. Isto possibilitará que o algoritmo possa ser utilizado para escrever um programa em qualquer linguagem de programação.

Exemplo1:

Elaborar um algoritmo que descreva as acções que são desencadeadas quando se pretende fazer uma sandes fiambre e queijo.

Resolução:

Exemplo 2:

Elabore um algoritmo em que descreva as acções que são desencadeadas quando se pretende efectuar um telefonema.

Resolução:

Exemplo 3:

Quando se prontifica para viajar no seu automóvel verifica que um pneu está furado. Elabore um algoritmo em que descreva as acções que são desencadeadas para resolver o problema.

Resolução:

# Estrutura de um programa em C

* A estrutura de um programa em C obedece a um determinado conjunto de regras denominadas sintaxe.
* A sintaxe da linguagem C, diferencia letras maiúsculas e minúsculas, isto é, “main”, “Main” e “MAIN” são termos distintos, pelo que se considera boa prática o uso de letras minúsculas para a formação de nomes de variáveis, funções e comandos.
* Os delimitadores **/\*** e **\*/**, identificam o inicio e o fim de comentários.
* { e } são os delimitadores de um bloco.
* A execução de um programa é feita a partir da execução da sua função principal main( ).

### Estrutura geral de um programa em C

/\* Definições de pré-processamento\*/

/\* Declaração de variáveis globais \*/

**main()**

**{** /\* declaração de variáveis locais\*/

/\* operações, chamadas de comandos e funções\*/

**}**

**Outras funções…..**

Palavras reservadas e identificadores predefinidos

**main ()**  - inicio do programa, da função

**{** – que indica o começo do programa ou de um conjunto de instruções;

**}** – que assinala o fim do programa ou de um conjunto de instruções;

Exemplo:

# include <stdio.h> /\* definição de pré-processamento necessário para o printf\*/

main()

{

printf(“ Bom dia…..Já acordaram?”);

}

As instruções em C são sempre encerradas por um ponto-e-virgula(;).

### Função printf()

É uma função de E/S (entrada e saída). Quando o programa encontra esta linha, imprime no ecrã o que estiver incorporada dela, neste caso:

“Bom dia…..Já acordaram?”

### Mudar a cor da letra e fundo ecrã

**System(“color 0f”);**

O comando color recebe dois números:

- O primeiro corresponde à cor do plano de fundo do ecrã

- O segundo corresponde à cor da fonte

Os números que devem ser digitados são números em hexadecimal (0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f)

### Limpar ecrã

**System(“cls”);**

### Pára o ecrã

Sleep(valor do tempo) -> vem em milissegundos

### Dois argumentos

**Exemplo:**

main ()

{

Printf(“Este e’ o numero dois: %d”,2);

}

Imprime no ecrã -> Este e’ o número dois:2

* No 1º exemplo temos um único argumento, agora temos dois argumentos separados por vírgula.
* A expressão de controle pode conter caracteres que serão exibidos no ecrã e códigos de formatação que indicam o formato em que devem ser impressos (%d solicita mostrar o segundo argumento em formato decimal).
* O número de argumentos é variável.

### Vários argumentos (cadeia de caracteres)

**Exemplo1**

main ()

{

**Printf(“%s** esta a %d milhoes de milhas \ndo sol”,”venus”,67);

}

**Exemplo2**

main ()

{

printf("A letra %c ",'j');

printf("pronuncia-se %s ","jota");

}

Note que o ‘j’ é um carácter e é delimitado por aspas simples enquanto que “jota” é uma cadeia de caracteres é delimitado por aspas duplas, isto implica que o compilador diferencie um carácter de uma cadeia de caracteres.

Observe também que a saída é feita em duas linhas de programa, o que não implica que sejam impressas em duas linhas, mas sim na mesma linha.

Tabela seguinte mostra os códigos de C para caracteres que não podem ser inseridos directamente do teclado:

|  |  |
| --- | --- |
| **Código** | **Significado** |
| \n  \t  \”  \\ | Nova linha  Tab  Aspas  Barra |

Tabela seguinte mostra os códigos de C para impressão formatada de printf()

|  |  |
| --- | --- |
| **Código** | **Significado** |
| %c  %d  %f  %s  %u | Carácter simples  Inteiro decimal  Ponto flutuante  Cadeia de caracteres  Decimal sem sinal |

## Variáveis

A programação em C exige a declaração do tipo das variáveis antes da sua utilização. A sua declaração possui a seguinte sintaxe:

**< tipo variável>** <lista de variáveis> **;**

Exemplo:

Int a,b,c;

Int a=2, b=5, c;

Os identificadores(variáveis) obedecem às seguintes regras:

* Começa com uma letra alfabética (maiúscula ou minúscula) ou underscore(\_).
* Os caracteres seguintes podem ser letras, algarismos e o underscore (\_).
* Não se pode usar espaços nem quaisquer outros caracteres, incluindo os caracteres acentuados.
* Maiúsculas e minúsculas são diferentes.
* Embora haja liberdade de escolher as designações que se quiser, é recomendável utilizar nomes sugestivos acerca do significado dos dados em causa.
* Não podem ter o mesmo nome das seguintes palavras reservadas na linguagem C (keywords) :

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

### Tipos de variáveis

O tipo de variável define a natureza dos valores que a variável pode ter.

Existem os seguintes tipos principais:

* **int** – valores numéricos inteiros positivos e negativos (-32768 a 32767), ocupa 2 bytes
* **float** – números reais em virgula flutuante (3.4E-38 a 3.4E+38), ocupa 4 bytes
* **double** – valores numéricos reais “longos” (1.7E-308 a 1.7E+308), ocupa 8 bytes
* **char** - carácter ASCII (-128 a 127) ocupa um byte (8 bits -> 28 256 combinações possíveis

Ex: ‘a’, ‘A’, ‘\*’

Existem qualificadores na linguagem C que podem ser utilizados com os tipos de dados atrás referidos por forma a aumentar ou alterar a sua gama de valores

|  |  |
| --- | --- |
| ***Qualificadores*** | ***Tipo onde se aplica*** |
| Short | Int |
| Long | Int, double |
| Unsigned | Char,int |

|  |  |
| --- | --- |
| Carácter | char (-128 a 127)  Unsigned char (0 a 255) |
| Inteiros | Int (-32768 a 32767)  Unsigned int (0 a 65535)------------unsigned  Short int (-128 a 127) -----------Short  Long int (-2.147.483.648 a 2.147.483.647) -----long  Unsigned short int (0 a 255)  Unsigned long int (0 a 4.294.967.295) |
| Vírgula flutuante | Float  Double  Long double 3.4x10 E-4932 a -3.4x10 E-4932 |

Exemplo:

main ()

{

int evento;

char corrida;

float tempo;

evento=5;

corrida=’C’;

tempo=27.25;

printf("O tempo na eliminatória %c ",corrida);

printf("\nda competicao %d foi %.2f.",evento,tempo);

}

### Inicializar variáveis

main ()

{

int evento=5;

char corrida=’C’;

float tempo=27.25;

printf("O tempo na eliminatoria %c ",corrida);

printf("\nda competicao %d foi %.2f.",evento,tempo);

}

### Variáveis ponto flutuante ( números reais)

* %.2f -> significa que a variável do tipo real fica com 2 casas decimais
* %6d -> significa que a parte inteira tem seis dígitos
* %4.2f ->significa que a parte inteira tem 4 dígitos e a decimal 2 dígitos
* %04d -> significa que a parte inteira tem 4 digitos, mas caso não tenha completa-a com zeros à esquerda (0021).
* Alinhamento à direita ou à esquerda:

main ()

{

printf("%10.2f %10.2f %10.2f\n”,8.0,15.3,584.13);

printf("%10.2f %10.2f %10.2f\n”,834.0,1500.55,4890.2);

}

main ()

{

printf("%-10.2f %-10.2f %-10.2f\n”,8.0,15.3,584.13);

printf("%-10.2f %-10.2f %-10.2f\n”,834.0,1500.55,4890.2);

}

1º programa -> alinhamento à direita

2º programa -> alinhamento à esquerda

## Caracteres gráficos

Todo o carácter (letra, digito, caracteres de pontuação, etc) é representado no computador por um número. O código ASCII dispõe de números de 0 a 255.

Os caracteres gráficos e outros não standard requerem uma outra maneira de escrita:

**\xdd**

Onde **dd** representa o código do carácter em hexadecimal.

Exemplo:

main ()

{

printf("\xdd\n”);

}

## Função scanf()

A função scanf() é outra função de E/S, ela é o complemento de printf() e permite-nos ler os dados da entrada padrão(teclado).

***Sintaxe:***

Scanf(“expressão de controle”, lista de argumentos)

* A expressão de controle pode conter códigos de formatação, precedidos por um sinal de %:

|  |  |
| --- | --- |
| **Código** | **Significado** |
| %c  %d  %f  %s  %u | Lê um único caracter simples  Lê um inteiro decimal  Lê um Ponto flutuante  Lê uma Cadeia de caracteres  Lê um Decimal sem sinal |

* A Lista de argumentos deve conter os endereços das variáveis, tendo um operador, chamado **operador de endereço (&)**, que retorna o endereço da variável.

### Operador de endereço (&)

O endereço de uma variável é o nome que o computador usa para identificar uma variável.

Toda a variável ocupa uma certa localização na memória e o seu endereço é o 1º byte ocupado por ela. Se declarar uma variável “**n**”como inteira e lhe atribuir o valor 2, quando for referenciada devolve 2, mas se for referenciada com **&n** devolve o endereço do primeiro byte onde “**n”** está guardada.

**Exemplo:**

main ()

{

int n;

n=2;

printf(“valor=%d, endereco=%u”,n,&n);

}

Saída no ecrã:

Valor=2, endereço=1370;

**Exemplo1 da função scanf()**

main ()

{

float anos, dias;

printf(“Digite a sua idade em anos:”);

scanf(“%f”,&anos);

dias=anos\*365;

printf("\nA sua idade em dias e’ %.0f.\n”,dias);

}

**Saída no ecrã:**

Digite a sua idade em anos:4

A sua idade em dias e’ 1460.0.

**Exemplo2 da função scanf()**

main ()

{

char a;

printf(“Digite um carácter e veja-o em decimal,”);

printf(“octal e hexadecimal.\n”);

scanf(“%c”,&a);

printf("\n%c=%d dec., %o oct. E %x hex. \n”,a,a,a,a);

}

**Saída no ecrã:**

Digite um carácter e veja-o em decimal, octal e hexadecimal.

m

m=109 dec., 155 oct. e 6D hex.

## Funções getche() e getch()

Em algumas situações a função scanf() não se adapta perfeitamente pois é necessário pressionar [enter] depois da entrada para que scanf() termine a leitura.

A função **getche()** lê o carácter do teclado no instante que é dactilografado sem esperar [enter], sendo este impresso quando é digitado.

**Exemplo da função getche()**

main ()

{

char ch;

printf(“Digite um carácter:”);

ch=getche();

printf("\n A tecla pressionada foi: %c ”,ch);

}

**Saída no ecrã:**

Digite um carácter: a

A tecla pressionada foi: a

A função **getch()** lê o carácter do teclado no instante que é dactilografado sem esperar [enter], não permitindo que este seja impresso quando é digitado.

**Exemplo da função getch()**

main ()

{

char ch;

printf(“Digite um caracter:”);

ch=getch();

printf("\n A tecla pressionada foi %c ”,ch);

printf("\ e a sua sucessora ASCII e’ %c \n”,ch+1);

}

**Saída no ecrã:**

Digite um carácter:

A tecla pressionada foi a e a sua sucessora ASCII e’ b

## Função getchar()

Obtém o próximo carácter de entrada cada vez que é chamada, e só termina a leitura quando é pressionada a tecla [enter].

## Função putchar()

Esta função é um complemento de getchar(), aceita um argumento cujo valor será impresso no ecrã.

**Exemplo da função getchar() e putchar()**

main ()

{

char ch;

printf("Digite um carácter:");

ch=getchar();

putchar(ch);

ch=getchar();

putchar(ch);

ch=getchar();

putchar(ch);

printf("\n");

}

Ou

main ()

{

char ch;

printf("Digite um carácter:");

putchar(getchar());

putchar(getchar());

putchar(getchar());

putchar(getchar());

printf("\n");

}

## Função gets()

Lê vários caracteres, equivalente ao getchar() e scanf(), mas suporta **espaços** vindos do teclado

main ()

{

char ch[100];

printf(“Digite um texto”);

//scanf(“%s”,&ch); não lê os espaços

gets(ch); // lê os espaços

printf("O texto digitado foi %s\n”,ch);

}

## Operadores

### Operadores Aritméticos

* + = (atribuição)
  + + (soma)
  + - (Subtracção)
  + \* (multiplicação)
  + / (Divisão)
  + % (resto da divisão)

#### Operador =

Representa a atribuição da expressão à direita ao nome da variável à esquerda.

Num=2000;

Nunca poderá fazer : 2000=num; visto que não podemos atribuir um valor a uma constante.

O C aceita várias atribuições numa mesma instrução:

Laranjas=cenouras=abacates=90;

#### Operadores + - / \*

Representam as operações aritméticas básicas e seguem as regras da matemática ( a multiplicação e a divisão tem prioridade sobre a subtracção e soma).

#### Operador %

Aceita somente operandos inteiros e retorna o resto da divisão:

Exemplo:

17%5=2

### Operadores de incremento (++) e decremento(--)

O operador incremento(++) incrementa o operando e trabalha em dois modos:

**Pré-fixado**

++n (n é incrementada antes do seu valor ser usado)

**Pós-fixado**

n++ (n é incrementada depois do seu valor ser usado)

exemplo:

main ()

{

Int n,x;

n=5;

x=n++

printf("x=%d n=%d \n”,x,n);

}

**Saída no ecrã:**

x=5 n=6

Vamos analisar as seguintes expressões:

k=3\*n++ -> PRIMEIRO n é multiplicado por 3

DEPOIS o resultado é atribuído a k

FINALMENTE n é incrementado de 1

k=3\*++n -> PRIMEIRO n é incrementado de 1

DEPOIS n é multiplicado por 3 FINALMENTE o resultado é atribuído a k

**Exemplo:**

main ()

{

int a,b,c,n;

a=2;

b=5;

n=(a+b++)\*3;

c=b+3

printf("n=%d b=%d c=%d \n”,n,b,c);

}

Substituindo os valores teremos:

n= (2+5)\*3=7\*3=21

b=6

c=9

### Operadores aritméticos de atribuição

* +=
* -=
* \*=
* /=
* %=

Cada um destes operadores é usado com um nome de variável à sua esquerda e uma expressão à sua direita.

Se **x** é uma variável, **exp** uma expressão e **op** um operador arirmético, então:

***x op=exp;***

**equivale a:**

**x=(x) op (exp);**

**exemplos:**

|  |  |  |
| --- | --- | --- |
| i +=2; | Equivalente a | i=i+2; |
| x\*=y+1; | Equivalente a | x=x\*(y+1); |
| t /=2.5; | Equivalente a | t=t/2.5; |
| p %=5; | Equivalente a | p=p%5; |
| d -=3; | Equivalente a | d=d-3; |

main ()

{

int total=0;

int cont=10;

printf("Total=%d\n”,total);

total+=cont;

printf("Total=%d\n”,total);

total+=cont;

printf("Total=%d\n”,total);

}

**Saída no ecrã:**

Total=0

Total=10

Total=20

### Operadores relacionais

* > (maior)
* < (menor)
* >= (maior ou igual)
* <= (menor ou igual)
* == (igual)
* != (diferente)

### Operadores Lógicos

* ! (negação)
* && (E)
* || (OU)

# Estruturas de controlo

Estruturas de decisão Com base numa condição 🡪 **IF…ELSE**

ou selecção

Com selector ou de escolha

Múltipla 🡪 **SWITCH…**

Com contador 🡪 **FOR…….**

Estruturas de repetição

ou ciclos

🡪 **WHILE…**

Com base numa

condição

🡪 **DO…WHILE**

## Estrutura de decisão ou selecção condicional “IF…ELSE”

**Sintaxe:**

**IF** (condição) <instrução/ões 1> **ELSE** <instrução/ões 2>

**IF** é a palavra reservada que assinala o inicio desta estrutura;

**(condição) -** é uma expressão do tipo booleano, isto é, que devolve como resultado um valor “verdade” (true) ou “falso” (false);

**<instrução/ões> -** representa uma determinada acção que se pretende efectuar; podendo ser uma instrução simples ou composta, no caso de a condição se verificar, ou seja, se a condição assumir o valor de “verdade”;

**ELSE –** é uma cláusula opcional da estrutura; se for usada, indica a acção ou acções a seguir, no caso da condição não se verificar, ou seja, assumir o valor “falso”;

**Diagrama:**

**F**

**V**

Condição

Instrução/ões 2

Instrução/ões 1

## Instruções múltiplas no corpo do comando IF

Em caso de várias instruções serem necessárias no corpo do comando IF, elas devem estar entre chavetas (inicio e fim do bloco).

Exemplo:

main ()

{

If ( getche() == ‘p’)

{

printf(“\n a tecla pressionada foi p.”);

printf(“\n Obrigada por escolher esta tecla”);

printf(“\n Ana Fernandes”);

}

else

{

printf(“Não pressionou a tecla p.”);

printf(“Para a próxima deverá carregar a tecla p”);

}

}

**Exercício :**

**Problema:**

{Efectue um algoritmo e respectivo programa para determinar o maior de dois números}

**Exercício :**

**Problema:**

{Efectue um algoritmo e respectivo programa para determinar o maior de três números}

**Exercício :**

**Problema:**

Efectue um algoritmo e respectivo programa para ler um número inteiro identificando o mês a que corresponde

**Exercício :**

**Problema:**

Elabore um algoritmo e respectivo programa para simular uma máquina de calcular. Deve pedir ao utilizador dois números inteiros, bem como a operação a efectuar (+,-,\*,/). Consoante a operação, os números deverão ser somados, multiplicados, subtraídos ou multiplicados, no final deverá mostrar o total .

**Exercício :**

**Problema:**

Elabore um algoritmo e respectivo programa para classificar as notas de um aluno. Deve pedir ao utilizador a nota que o aluno teve e mediante a nota deverá ser apresentada a sua classificação como mostra a seguinte tabela:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nota | >=0 e <4 | >=4 e <10 | >=10 e <13 | >=13 e <17 | >=17 e <=20 |
| Classificação | Mau | Medíocre | Suficiente | Bom | Muito Bom |

Deverá ainda colocar o fundo com a cor azul, bem como nas classificações, que deverão ter uma cor diferente para cada uma ao serem mostradas.

## Estrutura de decisão ou selecção condicional

**Sintaxe:**

**switch** <expressão inteira>

**{**

case constante1**:** instruções1;

case constante2**:** instruções2;

case constante3**:** instruções3;

…..

case constanten**:** instruçõesn;

**default:** instruções;

**}**

* O comando **switch** é similar ao if-else mas tem maior flexibilidade e formato claro.
* O comando switch avalia a expressão entre parênteses, após a palavra “switch”, e compara o seu valor com os vários casos existentes.
* Cada caso deve ser rotulado por uma constante do tipo inteiro ou carácter ou uma expressão constante.
* Pode haver uma ou mais instruções seguindo de cada case, estas instruções não devem estar entre chavetas.
* Se um caso for igual ao valor da expressão, a execução começa nele.
* O corpo do switch deve estar envolvido por chavetas.
* Se nenhum caso for satisfeito, existe um caso **default**: a execução começará nele, senão o programa processará as instruções seguintes ao bloco switch. O default é opcional.
* As constantes têm de ser todas diferentes.
* O comando break causa uma saída imediata do switch, se não existir o comando break, o programa segue todas as instruções dos casos.

## Estrutura de repetição ou ciclo

**Sintaxe:**

**do** **{**

instrução/instruções

**} while (expressão de teste);**

* O laço/ciclo **do-while** cria um ciclo repetido até que a condição (expressão de teste) seja falsa.
* Neste ciclo realiza-se 1º as instruções só depois é que se verifica a condição (expressão de teste).
* Este ciclo é realizado pelo menos uma vez.
* Este ciclo não tem um contador próprio.

**Diagrama:**

V

F

Instruções

Condição

**Exemplo**

**Problema:** Elabore um programa para efectuar a soma de 4 números inteiros a inserir a partir do teclado. No final deverá mostrar no ecrã o valor da soma.

**Modelo MENU**

int main()

{

char op;

do

{

system("cls");

printf("\n1 - Opcao1");

printf("\n2 - Sair\n");

op=getche();

switch (op)

{

case '1':

break;

case '2':printf("Obrigada por visitar o meu programa\n");

break;

default: printf("Não existe essa opcao");

sleep(3000);

}

}while (op!='2');

system("pause");

}

**Exercício**

**Problema:** Transforme o programa anterior de forma a efectuar a soma e a média de n números inteiros a inserir a partir do teclado. No final deverá mostrar no ecrã o valor da soma e da média.

Para tal efectue o seguinte menu:

1 – Quantidade de números

2 – Inserir números

3 – Mostrar soma

4 – Mostrar Média

5 - Sair

**Exercício**

**Problema:** Elabore um programa que indique a quantidade de elementos positivos e negativos de uma sequência de números reais a inserir a partir do teclado. Considere que a sequência termina com a introdução do número zero. No final além de indicar a quantidade de números inseridos, deverá indicar a soma dos valores positivos.

Para tal efectue o seguinte menu:

1 – Inserir números

2 – Mostrar quantidade

3 – Mostrar soma valores positivos

4 - Sair

**Exercício**

**Problema:** O programa que se segue é um jogo, ou seja, vai gerar um nº aleatório de 0 a 100 e pede ao utilizador que adivinhe o nº gerado. De cada que o utilizador indicar um nº o programa deverá informar se o nº gerado é maior ou menor que o nº introduzido. O programa deverá correr até ser encontrado o nº ou até um máximo de tentativas=10.

Para tal efectue o seguinte menu:

1 – Jogar

2 - Sair

**Exemplo gerar número aleatório**

int main()

{ int num;

srand(time(NULL));

num= rand() % 40;

printf("Numero Gerado: %d\n",num);

system("pause");

}

**srand –** inicializa de forma aleatória

**rand()-** gera o numero.

Ex.: rand() % max\_val

gera um numero de 0 a (max\_val-1)

## Estrutura de repetição ou ciclo

**Sintaxe:**

**while**  **(expressão de teste)**

**}**

instrução/instruções

**}**

* O laço/ciclo **while** é executado enquanto a condição (expressão de teste) for verdadeira.
* Neste ciclo verifica a condição (expressão de teste), só depois realiza as instruções.
* Este ciclo pode não ser realizado nenhuma vez.
* Este ciclo não tem um contador próprio.

**Diagrama:**

Condição

F

V

Instruções

**Exercício**

**Problema:** Elabore um programa que deverá pedir ao utilizador para inserir nºs até que o valor da soma desses nºs seja superior a 100. No final deverá mostrar no ecrã o valor da soma.

Para tal efectue o seguinte menu:

1 – Inserir números

2 – Mostrar soma

3 - Sair

**Exercício**

**Problema:** Elabore um programa que calcule a média de pesos de vários artigos, cujos valores serão introduzidos pelo utilizador. A série de valores a introduzir termina quando se tenha atingido um máximo de 10 artigos ou se o somatório dos pesos ultrapassar 5000.

Para tal efectue o seguinte menu:

1 – Inserir pesos

2 – Mostrar média

3 - Sair

**Exercício**

**Problema:** Elabore um programa para efectuar a soma de n números inteiros a inserir a partir do teclado. O utilizador deverá indicar a quantidade de nºs. Não deverá aceitar o valor 0(zero). No final deverá mostrar o maior valor inserido bem como a soma dos valores.

Para tal efectue o seguinte menu:

1 – Inserir Quantidade

2 – Inserir números

3 – Mostrar soma

4 – Mostrar o maior valor inserido

5 - Sair

**Exercício**

**Problema:** Vamos supor que nos são atribuídas 100 unidades monetárias para fazer apostas num jogo.

O jogo consiste no seguinte: de cada vez que se fizer uma jogada, o programa gera aleatoriamente um valor que pode ser 0 ou 1; se o valor obtido for 0, o jogador perde o que apostar; se for 1 ganha um valor igual ao da sua aposta.

O programa deve perguntar quanto queremos apostar e faze-lo dentro de um ciclo que dure enquanto o nº de jogadas não ultrapassar 10 e enquanto tivermos dinheiro.

Não podem ser aceites apostas superiores ao dinheiro disponível em cada momento.

Para tal efectue o seguinte menu:

1 – Jogar

2 - Sair

## Estrutura de repetição ou ciclo

**Sintaxe:**

**for**  **(inicialização; expressão de teste; incremento)**

**}**

instrução/instruções

**}**

* O laço/ciclo **for** engloba 3 expressões numa única.
* Os parênteses seguido da palavra for contêm três expressões separadas por pontos-e-virgulas.
* A **inicialização** é uma instrução de atribuição (conta=0) e é sempre executada uma única vez antes do laço ser iniciado.
* A **expressão teste** é uma instrução de condição que controla o laço(conta<10). Esta expressão é avaliada como verdadeira ou falsa toda vez que o ciclo for iniciado ou reiniciado. Se **verdadeira**, o corpo do ciclo é executado. Quando a expressão é falsa o ciclo é terminado e passa para a instrução seguinte ao laço.
* O **incremento** define a maneira como a variável de controle do ciclo será alterada cada vez que o ciclo é repetido(conta++). Esta instrução é executada, imediatamente após a execução do corpo do ciclo.
* O laço/ciclo **for** é útil principalmente quando queremos repetir algo um número fixo de vezes.

**Ordem ascendente** (imprime os números de 0 a 9)**:**

**int main()**

**{**

**Int conta;**

Inicialização de zero à variável conta

Incremento de 1 valor

teste

**For (conta=0; conta<10; conta++)**

**Não coloque ponto-e-virgula aqui**

**printf(“conta=%d\n”,conta);**

**system("pause");**

**}**

**Ordem descendente** (imprime os números de 9 a 0)**:**

**int main()**

**{**

**Int conta;**

Inicialização de nove à variável conta

decremento de 1 valor

teste

**For (conta=9; conta>=0; conta--)**

**Não coloque ponto-e-virgula aqui**

**printf(“conta=%d\n”,conta);**

**system("pause");**

**}**

**Ordem ascendente** (Imprime os números de 0 a 9 de 3 em 3)**:**

**int main()**

**{**

**Int conta;**

Incremento de 1 valor

Inicialização de zero à variável conta

teste

**For (conta=0; conta<10; conta+=3)**

**Não coloque ponto-e-virgula aqui**

**printf(“conta=%d\n”,conta);**

**system("pause");**

**}**

**Flexibilidade do ciclo For**

1. Qualquer uma das expressões de um ciclo for pode conter várias instruções separadas com vírgulas. A vírgula em C pode ser considerado um operador que significa “faça isto e isto”. Um par de expressões separadas por vírgulas é avaliado da esquerda para a direita.

Exemplo – Imprime os números de 0 a 98 em incremento de 2:

int main()

{ int x,y;

For (x=0, y=0; x+y<100 ; x=x+1, y=y+1)

printf(" %d",x+y);

system("pause");

}

1. Podemos usar caracteres em vez de inteiros

Exemplo – Imprime as letras minúsculas do alfabeto e seus respectivos códigos decimais da tabela ASCII:

int main()

{

char ch;

for (ch=’a’; ch<=’z’ ; ch++)

printf(" O valor ASCII de %c e´%d \n”,ch,ch);

system("pause");

}

1. Podemos usar chamadas a funções em qualquer uma das expressões do ciclo.

Exemplo – Este programa imprime a entrada de um texto. Lê caracter a caracter e imprime o caracter seguinte a partir do código ASCII .

int main()

{

char ch;

for (ch=getch(); ch!=’X’ ; ch=getch() )

printf("%c”,ch+1);

system("pause");

}

1. Qualquer uma das três partes do ciclo for pode ser omitida, embora os pontos-e-virgulas devam permanecer. Se a expressão de iniciação ou a de incrementação forem omitidas, elas serão simplesmente desconsideradas. Se a condição de teste não está presente é considerada permanentemente verdadeira.

Exemplo:

int main()

{

char ch;

for ( ; (ch=getch())!=’X’; )

printf(" %c”,ch+1);

system("pause");

}

1. O corpo do ciclo pode estar vazio, no entanto o ponto-e-virgula permanece.

Exemplo 1:

int main()

{

char c;

for ( ; (c=getch())!=’X’; printf(" %c”,c+1))

;

system("pause");

}

Exemplo 2:

int main()

{

char c;

for ( c=0; c<=100;c++)

;

system("pause");

}

**Exercício**

**Problema:** Elabore um programa que efectue o cálculo de uma potência, sendo dados pelo utilizador a respectiva base e expoente.

Para tal efectue o seguinte menu:

1 – Inserir base

2 – Inserir Expoente

3 – Calcular/Mostrar a potência

4 - Sair

**Exercício**

**Problema:** Elabore um programa que classifique as notas de uma turma. A quantidade de notas deverá ser introduzida pelo utilizador. O programa deverá além de classificar (positiva>=10) e (negativa<10), também deverá contabilizar o nº de notas positivas e negativas e calcular a nota média da turma. Ter em conta que o valor da nota tem de ser validado.

Para tal efectue o seguinte menu:

1 – Inserir quantidade

2 – Inserir/ classificar notas

3 – Quantidade de positivas

4 – Quantidade de negativas

5 – Média da turma

6 - Sair

# Arrays - Matrizes

**Arrays-** Vectores ou matrizes

**Conceito**

É um tipo de dados estruturado muito utilizado que pode agrupar numa mesma variável um conjunto de valores do mesmo tipo.

**Exemplo da representação gráfica de um array:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| I0 | I1 | I2 | I3 | I4 |
| 10 | 12 | 8 | 15 | 9 |

**Declaração de uma variável do tipo array**

**<tipo>** <nome\_array>[<intervalo do(s) índice(s)>]

**<tipo>** - Indica qual o tipo dos elementos colocados no array;

**<Nome\_array>** – Nome da variável;

**[<intervalo do(s) índice(s)>]** – Define o nº de elementos da variável( comprimento do array);

**Exemplo de uma declaração de um array**

int notas[10] ;

Neste exemplo declara-se um array com o nome ‘notas’, com 10 elementos que contém valores inteiros. Este array poderia ser substituído pela declaração de 10 variáveis do tipo inteiro, onde cada uma iria conter o valor de um elemento do array. Conclui-se então que a utilização de um array torna-se vantajoso em relação à utilização das outras variáveis.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| I0 | I1 | I2 | I3 | I4 | I5 | I6 | I7 | I8 | I9 |
| 10 | 12 | 8 | 15 | 9 | 12 | 11 | 14 | 15 | 10 |

**Instruções de leitura (a partir o teclado) de um array**

* **Leitura de um só elemento:**

printf(“Insira a nota do 3º aluno”);

scanf(“%d”,&notas[2])

**Nota:** O elemento referenciado pelo número 3 (notas[3]), não é o terceiro da matriz e sim o 4, pois a numeração começa em 0. Assim o ultimo elemento da matriz possui um índice, uma unidade menor que o tamanho da matriz.

* **Leitura de todos elementos:**

for (i=0;i<10;i++)

{

printf(“Insira a nota do %d aluno”,i+1);

scanf(“%d”,&notas[i])

}

**Instruções de escrita no ecrã de um array**

* **Escrita de um só elemento:**

printf(“ O valor da nota do aluno nº 3é: %d”, notas[2]);

* **Escrita de todos elementos:**

for (i=0;i<10;i++)

printf(“ O valor da nota do aluno nº %d é: %d”, i+1,notas[i]);

**Instruções de atribuição de valores a um array**

* Atribuição de um só elemento:

notas[3]=12;

**Exemplo:**

int main()

{

int notas[5];

int i,soma;

for (i=0;i<5;i++)

{

printf("digite a nota do aluno %d:",i+1);

scanf("%d",&notas[i]);

}

for (i=0, soma=0;i<5;i++)

soma=soma+notas[i];

printf("A media das notas: %d",soma/5);

system("pause");

}

**Exercícios práticos**

**Exercício**

**Problema:**

Pretende-se um programa que registe 6 temperaturas ocorridas diariamente numa localidade e que calcule a média dessas temperaturas. O programa deve também indicar quais as temperaturas que tiveram valores acima da média.

Deverá criar o seguinte menu:

* Inserir temperaturas
* Média temperaturas
* Temperaturas acima média
* Quantidade temperaturas superiores a 30
* Sair

**Exercício**

**Problema:**

Elabore um programa que determine tanto o máximo como o mínimo de um conjunto de n números, introduzidos pelo utilizador. Estes números são guardados na variável A como mostra o seguinte:

Deverá criar o seguinte menu:

* Quantidade de elementos a inserir(não deve ser maior do que tamanho do array)
* Inserir elementos
* Maior Elemento
* Menor Elemento
* Sair

**Exercício**

**Problema:**

Escreva um programa que registe os preços de 8 artigos diferentes e que, em seguida seja, capaz de actualizar esses mesmos preços sabendo que todos eles sofrem um aumento, consoante percentagem pedida ao utilizador.

Deverá criar o seguinte menu:

* Quantidade de elementos a inserir(não deve ser maior do que tamanho do array)
* Inserir preços
* Percentagem de aumento
* Listar preços actualizados
* Sair

**Exercício**

**Problema:**

Pretende-se um programa que, manipulando uma estrutura de dados **array**, permita a um distribuidor de computadores controlar as encomendas efectuadas pelos seus 20 clientes. O programa deve começar por afixar o seguinte menu de comandos:

|  |
| --- |
| **\*\*\*\*Menu\*\*\*\*\*** |
| **R – REGISTAR** encomenda  **A – ANULAR** encomenda  **M – MAIOR** encomenda  **L – LISTAGEM** das encomendas  **T – TOTAL** encomendo F – FIM |
| **OPÇÃO :\_\_** |

Ficando à espera que seja seleccionada uma das opções. Uma vez executadas as operações correspondentes à opção indicada deve voltar a apresentar o menu para que seja seleccionada nova opção.

**R -** O programa deverá pedir o numero do cliente. Se esse cliente já efectuou encomenda, deve apresentar uma mensagem de erro. Caso contrário deve pedir o valor encomendado e guardá-lo na respectiva posição do array.

**A –** O programa deverá pedir o número do cliente, procedendo de seguida à anulação da encomenda. Se esse cliente ainda não tinha efectuado nenhuma encomenda deve ser apresentada uma mensagem de erro.

**M –** Deve ser apresentado o valor da maior encomenda e o respectivo número de cliente.

**L –** Deve ser apresentada uma listagem de todos os clientes. Na listagem deve ser apresentado o número do cliente e o valor encomendado.

**T –** Deve ser apresentado o total encomendado.

**F –** Termina a execução do programa.

**Sugestão :** Considerando que os clientes estão numerados de 1 a 20, faça corresponder o número do cliente a uma posição do array