# Linguagem C

## 1- INTRODUÇÃO

A linguagem "C" foi criada na década de 70 por Dennis M. Ritchie a partir de uma outra linguagem: o "B", criada por Ken Thompson. O "B", por sua vez, veio da linguagem BCPL, inventada por Martin Richards.

A linguagem "C" tornou-se muito importante e popular por possuir tanto características de "alto nível[1] " quanto de "baixo nível[2] " sendo considerada uma linguagem de programação de "nível médio[3] ". Outra característica muito importante é ser portável, isto é, poder ser usada em várias máquinas de portes e sistemas operacionais diferentes.

A linguagem "C" possui a vantagem se ser estruturada e compilada sendo bastante usada para construção de sistemas operacionais, planilha eletrônica, interpretadores, editores de texto, compiladores, etc.

Com a evolução dos microcomputadores surgiram várias versões de compiladores "C" havendo a necessidade de padronização, formalizada pela ANSI.

## 2- ESTRUTURA BÁSICA DE UM PROGRAMA EM "C"

Um programa em "C" consiste em uma ou mais funções, sendo que a única função que necessariamente precisa estar presente é a denominada **main()** (principal), que é a primeira função a ser executada quando o programa se inicia. As outras funções podem proceder ou suceder a função main().

```
main( )
{
        // corpo da função – bloco de comandos e ou funções
}
```

Os parênteses após a palavra "main()", indica que é uma função. O nome das demais funções desenvolvidas em C, pode ser qualquer um, menos "main".

O código que estiver dentro das chaves será executado seqüencialmente quando a função for chamada. Cada bloco de instruções deve ser colocado entre chaves e podem ser aninhados uns dentro dos outros.

A linguagem "C" é "Case Sensitive", isto é, maiúsculas e minúsculas fazem diferença. Se for declarada uma variável com o nome soma ela será diferente de Soma, SOMA, SoMa ou sOmA. Da mesma forma que os comandos em "C" devem ser sempre escritos com letras minúsculas.

Os comentários e as observações do programa podem aparecer em qualquer

[1] Alto nível :- comandos com sintaxe próxima a linguagem humana e tipos de dados inteiro, real, caracter e string. Exemplo de linguagens de alto nível: Pacal, Delphi, Basic, Visual Basic, Clipper, etc.

[2] Baixo nível:- manipulação de bits, bytes e endereços. Exemplo: Assembler.

[3] Nível médio:- combina elementos das linguagens de alto nível com a funcionalidade das de baixo nível. Exemplo: Linguagem "C".

lugar desde que colocados entre os delimitadores /* comentário */ ou // comentário. Em “C” as instruções são sempre encerradas por “; “.

## TIPOS BÁSICOS DE DADOS

| Tipo | Bit | Byte | Escala |
|---|---|---|---|
| char | 8 | 1 | -128 a 127 |
| int | 16 | 2 | -32768 a 32767 |
| float | 32 | 4 | 3.4E-38 a 3.4E+38 |
| double | 64 | 8 | 1.7E-308 a 1.7E+308 |
| void | 0 | 0 | sem valor |

## MODIFICADORES DE TIPO

Exceto o void, os tipos de dados básicos podem ter vários modificadores precedendo-os. Um modificador é usado para alterar o significado de um tipo básico para adaptá-lo mais precisamente às necessidades de diversas situações:

signed (com sinal)
unsigned (sem sinal)
long (máxima precisão)
short (menor precisão)

| TIPO | TAM.(bytes) | ESCALA |
|---|---|---|
| Char | 1 | -127 a 127 |
| Unsigned char | 1 | 0 a 255 |
| Signed char | 1 | -127 a 127 |
| Int | 2 | -32767 a 32767 |
| Unsigned int | 2 | 0 a 65535 |
| Signed int | 2 | -32767 a 32767 |
| Short int | 2 | -32767 a 32767 |
| Unsigned short int | 1 | 0 a 65535 |
| Signed short int | 1 | -32767 a 32767 |
| Long int | 4 | -2147483647 a 2147483647 |
| Signed long int | 4 | -4294967295 a 4294967295 |
| Unsigned long int | 4 | 0 a 4294967295 |
| Float | 4 | $3.4 *(10^{-38})$ a $3.4 * (10^{+38})$ |
| Double | 8 | $1.7*(10^{-308})$ a $1.7*(10^{+308})$ |
| Long double | 16 | $3.4*(10^{-4983})$ a $3.4*(10^{+4983})$ |

**- IDENTIFICADORES**

São usados para dar nomes às variáveis, constantes, tipos e ou funções. Para a criação desses identificadores deverá ser consideradas as informações a seguir:

- Podem ser formados por letras, números ou sublinhado(_);
- Tem que começar por letra ou sublinhado;
- Letras maiúsculas e minúsculas são caracteres distintos;
- Não podem ser palavras reservadas;
- Podem conter qualquer tamanho porém somente os 31primeiros caracteres são significativos.

**-VARIÁVEL**

É uma posição de memória com um nome, que é usada para armazenar uma informação de um tipo específico que pode ser modificada pelo programa.

Forma Geral :-

**tipo lista_de variáveis;**

Exemplo:-
```
int a,b,d;
float x,y;
unsigened int w;
```

**-Inicialização de variáveis:**

As variáveis podem ser inicializadas no mesmo momento em que elas são declaradas, colocando um sinal de igual e a informação desejada.

Exemplo:
```
char op='S'; /* as informações do tipo caracter são envolvidas por apóstrofes */
int primeiro=0;
char mensagem[20]="Bom Dia !!"; /* as strings são envolvidas por aspas */
```

**-MODIFICADORES DE TIPO DE ACESSO**

- **Constantes :-** Não podem ser modificadas pelo programa.

Forma Geral:-

**const tipo identificador;**

Exemplo:-
```
const int max=10;
const float taxa1=0.7,taxa2=0.5;
```

Outra forma de declarar uma informação como constante é usando a diretiva de compilação **#define.**

Forma Geral:-

**#define identificador=informação** /* não tem ; */

Exemplo:- #define volume=10
#define pi=3.141516

- **Volatile :-** É usado quando uma variável pode ser modificada inesperadamente por eventos externos ao controle do programa.

Forma Geral:-
**volatile tipo lista_de_variáveis;**

Exemplo:- volatile int tempo; /* pode ser modificada pelo hardware para atualizar a hora */.

**– OPERADORES**

- **Atribuição:-** nome_da_variável **=** expressão;

Exemplo:- x=2;
sexo='F';
total=(a*b)/100;
x=y=z=0; /* pode-se atribuir o mesmo valor a muitas variáveis */

**Abreviações:-**

| | | |
|---|---|---|
| x=x+100; | ⇒ | x+=100; |
| b=b*7; | ⇒ | b*=7; |
| z=z-2; | ⇒ | z-=2; |

- **Aritméticos:-** - subtração
+ adição
* multiplicação
/ divisão /* quando a divisão for inteira o dado é truncado */
% resto da divisão de números inteiros
++ incremento de 1 /* equivale x=x+1 */
--decremento de 1 /* equivale x=x-1 */

**OBS:-** Os operadores de incremento e decremento podem ser usados como prefixo ou sulfixo da variável sobre a qual estão sendo aplicados. A diferença é que quando são prefixo eles incrementam e retornam o valor da variável já incrementada. Quando são sulfixo eles retornam o valor da variável sem o incremento e depois incrementam a variável.

Exemplos:-
X=23;
Y=X++;
Resultado é Y=23 e X=24.

X=23;
Y=++X;
Resultado é Y=24 e X=24.

X=5;Y=2;

```
Z=X/Y;
Resultado é   Z=2

X=1;Y=2;
Z=X%Y;
Resultado é Z=1
```

- **Relacionais:**

```
> maior que
< menor que
>= maior ou igual
<= menor ou igual
== igual
!= diferente
```

- **Lógicos:**

```
&&    (e)
||      (ou)
!       (não)
```

Existem outros operandos que não serão tratados por enquanto. Exemplo: operadores de ponteiros, operador ponto, operador seta, etc.

- **Precedência dos Operadores**

| | |
|---|---|
| maior | ! ++ -- |
| | * / % |
| | + - |
| | < > <= >= |
| | == != |
| | && |
| | \| \| |
| menor | = |

Os operadores do mesmo nível de precedência são avaliados pelo compilador da esquerda para a direita. Os parênteses podem ser usados para alterar a ordem.

Em C, VERDADEIRO é qualquer valor diferente de zero e o FALSO=0. As expressões que usam operadores relacionais ou lógicos devolvem zero para falso e 1 para verdadeiro.

**3 - FUNÇÕES DE TELA** (biblioteca conio.h)

**clrscr() -** limpar a tela;
**clreol() -** limpa a linha;
**gotoxy(coluna,linha) -** posiciona o cursor;
**window(coluna incial,linha inicial,coluna final, linha final) -**cria uma janela na tela;
**textcolor(cor)** - seleciona a cor dos caracteres de texto;
**textbackground(cor)** – seleciona a cor do fundo da tela;

***TABELA DE CORES:***

| | |
|---|---|
| 0 | **PRETO** |
| 1 | **AZUL** |
| 2 | **VERDE** |
| 3 | **CIANO** |
| 4 | **VERMELHA** |
| 5 | **MAGENTA** |
| 6 | MARROM |
| 7 | CINZA CLARO |
| 8 | CINZA ESCURO |
| 9 | AZUL CLARO |
| 10 | VERDE CLARO |
| 11 | CIANO CLARO |
| 12 | ALARANJADO |
| 13 | MAGENTA CLARO |
| 14 | AMARELO |
| 15 | BRANCO |

**4- FUNÇÕES DE ENTRADA E SAÍDA DE DADOS** (biblioteca <stdio.h>)

**getchar() -** entrada de um caracter individual;
**getch() -** entrada de um caracter. O caracter não é exibido na tela não é necessário teclar <enter>;
**getche() -** entrada de um caracter. O caracter é exibido na tela não é necessário teclar <enter>;
**putchar -** exibe um caracter na tela;
**gets() -** entrada de string;
**puts() -** saída de string;
**scanf() -** entrada de dados formatada com string de controle de acordo com o tipo da variável;
**printf() -** saída de dados formatada com string de controle de acordo com o tipo da variável;
**cprintf() -** saída de dados formatada colorida com string de controle de acordo com o tipo da variável;

**- CONSTANTES BARRA INVERTIDA**

São utilizadas para controlar a exibição de dados na tela e na impressora.

| CÓDIGO | SIGNIFICADO |
|---|---|
| \n | nova linha (line feed) |
| \b | retrocesso (backspace) |
| \t | tabulação horizontal |
| \v | tabulação vertical |
| \a | alerta(beep) |
| \” | para exibir aspas |
| \\ | para exibir barra invertida |
| \f | salto de folha (form fee) |
| \r | retorna ao início da linha (return) |
| \0 | nulo |

**- COMANDOS DE FORMATO**

São constantes que definem o tipo e o formato dos dados a serem exibidos.

| **Comando** | **Formato** |
|---|---|
| %c | caracter |
| %d | inteiros decimais com sinal |
| %i | inteiros decimais com sinal |
| %f | decimais com ponto flutuante |
| %s | string |
| %e | notação científica |
| %o | octal |
| %x | hexadecimal |
| %l | inteiro longo |
| %lf | double |
| %u | decimal sem sinal |

**- A FUNÇÃO printf()**

A função printf serve para exibir uma informação no vídeo ou na impressora e possui a seguinte estrutura:

Forma Geral:

**printf**(“expressão de controle”, lista de argumentos);

Exemplo1:

```
int i=4;
float=5.5;

printf(“%d\n”, i);

printf(“Este numero e inteiro: %d    e este e real: %f”, i , j);
```

Resultado: 4

Este número é inteiro: 4 e este é real: 5.5

Exemplo2:

```
printf(“%s esta a %d milhões de milhas do sol”, “Venus”, 67);
```

Resultado: Venus esta a 67 milhões de milhas do sol

**- Diferenças entre impressão de caracter e cadeia de caracteres:**

printf(“A letra %c pronuncia-se %s”, ‘j’, “jota”);

Resultado: A letra j pronuncia-se jota

**-Tamanho de campos de impressão**

Vamos abordar este assunto através de exemplos:

```
Exemplo1:
main()
{
        printf(“Os alunos são %2d\n”,350);
        printf(“Os alunos são %4d\n”,350);
        printf(“Os alunos são %6d\n”,350);
}
```

Os números 2, 4 e 6, indicam o tamanho do campo de impressão e \n indica saltar para outra linha.

Resultado:

```
Os alunos são 350
Os alunos são   350
Os alunos são       350
```

Exemplo2:

```
main()
{
        printf(“%4.2f\n”,3456.78);
        printf(“%3.2f\n”,3456.78);
        printf(“%3.1f\n”,3456.78);
        printf(“%10.3f\n”,3456.78);
}
```

Resultado:

```
3456.78
3456.78
3456.8
     3456.780
```

**- Complementando com zero à esquerda**

```
main()
{
        printf(“\n%04d”,21);
        printf(“\n%06d”,21);
        printf(“\n%6.4d”,21);
```

```
        printf("\n%6.0d",21);
}
```

Resultado:

```
0021
000021
    0021
        21
```

**- Imprimindo caracteres**

Em C um caractere pode ser representado de diversas maneiras: o próprio caractere entre aspas simples ou sua representação decimal, hexadecimal ou octal segundo a tabela ASCII.

Exemplo:

```
printf(" %d   %c   %x   %o \n",'A', 'A', 'A', 'A');
```

Resultado:

```
65   A   41   101
```

**- A FUNÇÃO scanf()**

A função scanf() é outra das funções de E/S implementadas em todos os compiladores C. Ela é o complemento de printf() e nos permite ler dados formatados da entrada padrão (teclado).

Forma Geral:

**scanf**("expressão de controle", lista de argumentos)

A lista de argumentos deve consistir nos endereços das variáveis. C oferece um operador para tipos básicos chamado operador de endereço e referenciado pelo símbolo & que retorna o endereço do operando. Na função scanf() cada nome de variável deve ser precedida por um ampersand(&) ou E-comercial. Somente strings não devem conter & na frente.

Exemplo1: O programa a seguir exibe a idade em dias de uma pessoa.

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

void main()
{
        int anos, dias;
        printf("Digite sua idade em anos: ");
        scanf("%d",&anos);
```

```
        dias = anos*365;
        printf("Sua idade em dias é %.d\n",dias);
        getchar();
}
```

Resultado:

```
        Digite sua idade em anos: 4
        Sua idade em dias é     1460
```

Exemplo2: O programa abaixo transforma a temperatura de graus fahrenheit para graus celsius.

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

void main()
{
        float ftemp,ctemp;
        clrscr();
        printf("Digite temperatura em graus Fahrenheit: ");
        scanf("%f,&ftemp);
        ctemp=(ftemp-32) * 5/9;
        printf("Temperatura em graus Celsius é %f", ctemp);
        getchar();
}
```

Exemplo3: O programa abaixo calcula a média aritmética de duas notas de um aluno.

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

void main()
{
        float media,nota1,nota2;
        clrscr();
        printf("\t Digite a 1ª nota: ");
        scanf("%f",&nota1);
              printf("\t Digite a 2ª nota: ");
              scanf("%f",&nota2);
        media=(nota1 + nota2)/2;
        printf("\t A média é: %5.2f", media);
        getch();
}
```

Para segurança não utilize outras funções de entrada em um programa que está utilizando scanf().

O comando scanf poderá armazenar uma string desde que não contenha espaço em branco. O comando apropriado para entrada de dados do tipo string é o gets( ).

## 5- COMANDOS DE SELEÇÃO

Uma estrutura de seleção permite que uma ou mais ações sejam executadas a partir do resultado lógico de uma ou mais condições.

Forma Geral:-

```
If  (expressão) comando1;
else  comando2;
```

onde: expressão: - expressão lógica ou relacional ou chamada à uma função;
comando: - qualquer comando válido da linguagem “C”.

O comando **if** representa uma tomada de decisão que será resultado da expressão. Se o resultado for diferente de zero o comando1 será executado caso contrário ( se o resultado for igual a zero) será executado o comando2.
Para delimitar blocos de comandos utiliza-se { }.

```
If  (expressão)
    {
       comando1;
       comando2;
       ....
    }
else
    {
        comando1;
       comando2;
       ....
    }
```

Exemplos:-

```
If      (x>9)       y=100;
else                 y=200;

If ((x==2) && (x!=1))
{
        a=10;
        b=20;
        c=30;
}
else
        a=b=c=0;
```

Pode-se usar o **operador ?** para substituir o comando **if else.**

Forma Geral:-

Expressão1 **?** Expressão2 **:** Expressão3;