1. Introdução

A linguagem C foi primeiramente criada por Dennis M. Ritchie e Ken Thompson nos laboratórios da empresa "Bell", em 1972. C foi baseada na linguagem B de Thompson, que por sua vez era uma evolução da linguagem BCPL. Esta linguagem foi inicialmente concebida para ser utilizada no sistema operacional Unix.

O C é uma linguagem de programação genérica que é utilizada para a criação de programas diversos como processadores de texto, planilhas eletrônicas, sistemas operacionais, programas de comunicação, programas para a automação industrial, gerenciadores de bancos de dados, programas para a solução de problemas da Engenharia, Física, Química e outras Ciências, etc.

2. Características

- O C é uma linguagem de alto nível com uma sintaxe bastante estruturada e flexível tornando sua programação bastante simplificada.
- Programas em C são compilados, gerando programas executáveis.
- O C compartilha recursos tanto de alto quanto de baixo nível, pois permite acesso e programação direta do microprocessador. Com isto, rotinas cuja dependência do tempo é crítica, podem ser facilmente implementadas usando instruções em Assembly. Por esta razão o C é a linguagem preferida dos programadores de aplicativos.
- O C é uma linguagem estruturalmente simples e de grande portabilidade. O compilador C gera códigos mais enxutos e velozes do que muitas outras linguagens.

2.1 O C é "Case Sensitive"

Um ponto de suma importância: o C é "Case Sensitive", isto é, *maiúsculas e minúsculas fazem diferença*. Se declararmos uma variável com o nome soma ela será diferente de **Soma**, **SOMA**, **SoMa** ou **sOmA** . Da mesma maneira, os comandos do C **if** e **for**, por exemplo, só podem ser escritos em minúsculas pois senão o compilador não irá interpretá-los como sendo comandos, mas sim como variáveis.

3. Declaração de variáveis e constantes

3.1 - Declaração de variáveis

Sintaxe: tipo lista_variáveis;

tipo: deve ser um tipo de dado válido.
lista_variáveis: um ou mais identificadores separados por vírgula.

Exemplo:

```
main()
{
 int i,j,k;
 float a,b;
 char ch;
```

5. Operadores

É um símbolo que obriga o compilador a executar determinadas operações matemáticas, comparativas ou lógicas.

5.1 - Operadores aritméticos

Operador	Ação
-	subtração
+	adição
*	multiplicação
/	divisão
%	resto da divisão
--/++	decremento/incremento

Precedência (Hierarquia nas operações)

Hierarquia	Operação
1	Parênteses
2	Função
3	++ --
4	- (menos unário)
5	* / %
6	+ -

Observação: Quando houver duas ou mais operações de mesma ordem o computador executa da ***ESQUERDA*** para a ***DIREITA***.

5.2 - Operadores Relacionais

Operador	Ação
>	maior que
>=	maior ou igual a

<	menor que
<=	menor ou igual
==	igual a
!=	diferente de

5.3 - Operadores lógicos

Operador	**Ação**
&&	AND (e)
||	OR (ou)
!	NOT (não)

As expressões que usam operadores relacionais e lógicos retornam 0 (zero) para falso e !0 (não zero para verdadeiro), ou seja:

5.5 - Operador de atribuição

O operador de atribuição é o sinal de igual "**=**".

C permite usar o operador de atribuição em expressões que também envolvem outros operadores.

```
if ((produto = x * y) < 0)
```

- Primeiro "C" atribui o valor x * y ao produto, para depois avaliar a expressão.

6. Funções de Entrada e Saída

6.1 **A função *printf()*** – serve para escrever mensagens e conteúdo de variáveis (saída de dados).

A função **printf()** tem a seguinte forma geral:

printf (string_de_controle,lista_de_argumentos);

A string de controle mostra não apenas os caracteres que devem ser colocados na tela, mas também quais as variáveis e suas respectivas posições. Isto é feito usando-se os códigos de controle, que usam a notação **%**. Alguns dos códigos **%**:

Código	**Significado**
%d	Inteiro
%f	Float

%c	Caractere
%%	Coloca na tela um %

Além dos códigos de controle, existem também os códigos especiais, tais como o '\n' que já vimos anteriormente. Abaixo listamos estes códigos especiais:

Código	**Significado**
\n	Nova Linha

6.2 **A função *scanf()*** - serve para fazer a leitura de dados (entrada de dados).

O formato geral da função **scanf()** é:

scanf (string-de-controle,lista-de-argumentos);
Usando a função **scanf()** podemos pedir dados ao usuário. Mais uma vez, devemos ficar atentos a fim de colocar o mesmo número de argumentos que o de códigos de controle na string de controle. Outra coisa importante é lembrarmos de colocar o **&** antes das variáveis da lista de argumentos.
Exemplo: scanf("%d" , &num); faz a leitura de uma variável do tipo inteiro chamada num.

7. Comandos de Seleção

7.1 Comando if ()

Sintaxe:

```
if (condição)                          if (condição)
      comando 1;          ou               comando;
else
      comando 2;                                  // COMANDO SIMPLES
```

"else é opcional. Se a condição for avaliada como verdadeiro (qualquer valor diferente de 0), comando 1 será executado, caso contrário (condição falsa, valor igual a zero) comando 2 será executado. Comando 1, comando 2 ou comando podem ser simples ou compostos, ou seja, quando há mais de um comando ligado a outro, usa-se chaves ({ }), da seguinte maneira:

```
if (condição)                  if (condição)
  {                              {
    comando 1;        ou         comando 1;
    comando 2;                   comando 2;
  }                              comando 3;     // COMANDO COMPOSTO
else                           }
```

comando 3;

Exemplo:

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

main( )
{
 int numero;

printf("Entre com um numero inteiro: ");
 scanf("%d",&numero);
 if ((numero % 2) == 0)
     printf("NUMERO PAR\n");                  // Comando Simples
 else
     printf("NUMERO IMPAR\n");    // Comando Simples
 getch();
}
```

7.3 Comando switch ()

Sintaxe:

```
switch (variável)
        {
          case constante_1 : seqüência de comandos;
                             break;

          case constante_2 : seqüência de comandos;

                             break;

                    .

                  .
                  .
          default: seqüência de comandos;
        }
```

O computador testa uma variável sucessivamente contra uma lista de constantes inteiras ou caracteres. Depois de encontrar uma coincidência, o programa executa o comando ou blocos de comandos que estejam associados àquela constante. O comando

default é executado se não houver nenhuma coincidência. O comando break é usado para obrigar a saída do comando switch.

Exemplo:

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

main()
{
 int digito;

printf("Entre com um digito [0 .. 9]: ");
 scanf("%d",&digito);
 switch (digito)
        {
         case 0: printf("Zero\n");
                  break;
         case 1: printf("Um\n");
                  break;
         case 2: printf("Dois\n");
                  break;
                .
                .
                .
         case 9: printf("Nove\n");
                   break;
        default: printf("ERRO: Não é digito\n");
       }
 getch();
}
```

Exemplo:

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

int main()
{
 char digito;

printf("Entre com um digito [0 .. 9]: ");
 scanf("%c",&digito);
 switch (digito)
        {
         case '0': printf("Zero\n");
                    break;
```

```
        case '1': printf("Um\n");
                  break;
        case '2': printf("Dois\n");
                  break;
           .
           .
           .
        case '9': printf("Nove\n");
                  break;
       default: printf("ERRO: Não é digito\n");
      }
 getch();
}
```