Introdução à Programação C

Exemplo de um Programa

```
// Exemplo de programa em C
// Isto é uma linha de comentário
void main()
{
   int a;            // declara a variável "a"
   a = 3 + 2;        // soma 3 com 2
}
```

- Um programa em C é composto por um conjunto de **Funções**. A função pela qual o programa começa a ser executado chama-se ***main***.
- Após cada cada comando em C deve-se colocar um ; (ponto-e-vírgula).
- Um programa em C deve ser **Identado** para que possa ser lido com mais facilidade.

Identificadores

São os nomes que podem ser dados para variáveis e funções.

Para a escolha destes nomes é necessário seguir alguams regras:

- Um identificador deve iniciar por uma letra ou por um "_" (*underscore*);
- A partir do segundo caracter pode conter letras, números e *underscore*;
- Deve-se usar nomes significativos dentro do contexto do programa;
- C é uma linguagem *case-sensitive*, ou seja, faz diferença entre nomes com letras maiúsculas e nomes com letras minúsculas. `Peso` e `peso` são diferentes;
- Costuma-se usar maiúsculas e minúsculas para separar palavras: "`PesoDoCarro`";
- Deve ser diferente dos comandos da linguagem;
- Deve ter no máximo 31 caracteres (no caso do TurboC);
- Pode conter números a partir do segundo caracter;
- Exemplos:

```
Idade, Contador, PesoDoCarro,
Usuario_1, CorDaPagina, RaioDoCirculo
```

Variáveis

Uma variável é uma possição de memória que pode ser identificada através de um nome.

Podem ter seu conteúdo alterado por um ***comando de atribuição***.

Após a atribuição mudam de valor.

```
int a,b, SomaGeral;

a = 3; // a recebe o valor 3

b = a * 2; // b recebe o dobro do valor de a

c = a + b + 2; // c recebe 11
```

Tipos de Variáveis

- Todas as variáveis em C tem um ***tipo***;
- Cada tipo define os valores que a variável pode armazenar;
- Cada tipo ocupa uma certa quantidade de nemória.

Tipo	Tamanho	Valores Válidos
char	1 byte	letras e símbolos: 'a', 'b', 'H', '^', '*','1','0'
int	2 bytes	de -32767 até 32767 (apenas números inteiros)
float	4 bytes	de -3.4×10^{38} até $+3.4 \times 10^{+38}$ com até 6 dígitos de precisão
double	8 bytess	de -1.7×10^{308} até $+1.7 \times 10^{+308}$ com até 10 dígitos de precisão

Declaração de Variáveis

- Todas as variáveis **tem que ser declaradas *antes*** de serem usadas;
- Não há uma inicialização implícita na declaração

```
// Exemplo de programa em C

#include <stdio.h>   //  Arquivo de cabeçalho (header)
void main()
{
   int contador;            // declarações simples
   float PrecoDoQuilo;
   double TaxaDeCambio;
   char LetraDigitada;
   int IdadeManoel, IdadeJoao, IdadeMaria;  // Pode colocar mais de uma variável na
                                            // na mesma linha

   double  TaxaDoDolar,
           TaxaDoMarco,
           TaxaDoPeso,        // Também pode trocar de linha no meio
           TaxaDoFranco;

.......
}
```

Inicialização de Variáveis na Declaração

```
// Exemplo de programa em C

#include <stdio.h>   //  Arquivo de cabeçalho (header)
void main()
{
   int NroDeHoras = 0;            // declara e inicializa com Zero
   float PrecoDoQuilo = 10.53;    // declara e inicializa com 10.53
   double  TaxaDoDolar = 1.8,
           TaxaDoMarco = 1.956,
           TaxaDoPeso = 1.75,
           TaxaDoFranco = 0.2;

.......
}
```

Constantes

Constantes são identificadores que não podem ter seus valores alterados durante a execução do programa.

Para criar uma constante existe o comando `#define` que, em geral é colocado no início do programa-fonte.

Exemplos

```
#define LARGURA_MAXIMA 50           // Não se coloca ponto-e-vírgula após o valor
#define NRO_DE_DIAS_DA_SEMANA 7
#define NRO_DE_HORAS_DO_DIA 24
#define VALOR_DE_PI  3.1415

void main ()
{
   int TotalDeHoras;

    TotalDeHoras = 10 * NRO_DE_DIAS_DA_SEMANA * NRO_DE_HORAS_DO_DIA;
......
}
```

Strings

Uma String é uma seqüência de caracteres entre aspas duplas: ***"exemplo de uma string em C"***.

A função *printf*

A função `printf` exibe um ou mais dados na tela. Para tanto ele deve receber pelo menos dois parâmetros, separados por vírgula:

- um string de formato que define, através de caracteres especiais, os tipos dos dados a serem impressos e suas posições na linha de impressão;
- um dado a ser impresso. Este dado pode ser qualquer um dos dados visto anteriormente.

Por exemplo:

```
printf("%s","teste");
```

"%s" : é a string de formato
"teste" : é o dado a ser impresso.

A ***string de formato*** define quais os tipos dos dados a serem impressos. O símbolo **%s** será substituído pelo dado que vem após a vírgula.
Os **dados** definem quais os valores a serem impressos.

Se for necessário, um string de formato pode definir que mais de um dado será impresso. Para tanto, dentro da string de formato deve haver mais de um **%**, um para cada dado a ser impresso.
Neste caso, os dados devem vir após a string de formato separados por vírgulas.

Por exemplo:

```
printf("%s %s","teste1", "outra string");
```

Isto irá imprimir o string **teste1** deixar 1 espaço em branco e imprimir ao lado o string **outra string**, assim :

```
teste1 outra string
```

```
#include <stdio.h>      // Necessário para usar a função printf
                        // A função printf exibe um ou mais dados na tela

void main ()
{
   printf("%s","Isto é uma string ....\n"); // note o '\n' no final da string;
```

```
   printf("%s","Outra string ....");
   printf("%s","Terceira string\n");

//Depois de Executar o programa, tecle ALT-F5 para ver o resultado na tela
}
```

Exercício

Experimente colocar um '**\n**' entre os **%s** na string de formato.

```
printf("%s\n%s","teste1", "outra string");
```

Inclusão de Texto na String de Formato

É possível incluir um texto dentro da string de formato. Este texto irá aparecer exatamente como for digitado no programa-fonte.

O exemplo

```
printf("A aluna %s ficou doente","Maria");
```

geraria

```
A aluna Maria ficou doente
```

como resultado.

Constantes do tipo String

```
#define UNIVERSIDADE "Pontifícia Universidade Católica do Rio Grande do Sul"
        // deve-se colocar entre aspas

#include <sdtio.h>
#include <conio.h>   // necessário para as funções clrscr e getch

void main ()
{
   clrscr();     // Limpa a tela
   printf("%s", UNIVERSIDADE); // Imprime o nome representado pela constante
   getch();      // espera que o usuário pressione uma tecla
}
```

Impressão de Inteiros com *"printf"*

Para imprimir um inteiro com ***printf*** usa-se o símbolo `%d`

```
// Impressão de Variáveis Inteiras
#include <stdio.h>
#include <conio.h>   // necessário para as funções clrscr e getch

void main ()
{
   int Contador;
   int NroDeFilhos;

   clrscr();     // Limpa a tela

   Contador = 10;
   printf("Valor da Variável: %d\n", Contador);         // No momento da execução sinal %d vai
                                                 // ser substituído pelo valor da
                                                 // variável Contador

   NroDeFilhos = 3;
   printf("Maria tem %d filhos", NroDeFilhos);  // o inteiro pode ficar no meio da string
```

```
   getch();      // espera que o usuário pressione uma tecla
}
```

Impressão de Expressões aritméticas

```
// Impressão de Expressões aritméticas
#include <stdio.h>
#include <conio.h>   // necessário para as funções clrscr e getch

void main ()
{
   int NroDeAndares;
   int AlturaPorAndar;

   clrscr();     // Limpa a tela

   NroDeAndares = 7;
   AlturaPorAndar = 3;

   printf("Altura Total do Prédio: %d metros", NroDeAndares*AlturaPorAndar);
        // No momento da execução sinal %d vai ser substituído
        // pelo valor da multiplicação

   getch();      // espera que o usuário pressione uma tecla
}
```

Impressão de Números reais

```
// Impressão de números reais
#include <stdio.h>
#include <conio.h>   // necessário para as funções clrscr e getch

void main ()
{
   float NotaDaP1, NotaDaP2;
   float Media;


   clrscr();     // Limpa a tela

   NotaDaP1 = 6.6;  // Atribuição do Valores das médias
   NotaDaP2 = 8.2;

   Media = (NotaDaP1 + NotaDaP2) / 2.0;

   printf("Média Final : %f", Media);
        // No momento da execução sinal %f vai ser substituído
        // pelo valor da variável Media com SEIS casas decimais
        // Média Final : 7.400000
   getch();      // espera que o usuário pressione uma tecla
}
```

Formato de Impressão dos Números Reais

No exemplo acima o resultado da média (7.4) foi impresso com 6 casas decimais (7.400000).

Isto sempre acontece quando se manda imprimir um ***float*** da forma como foi feito no exemplo acima. Isto acontece pois o padrão da função ***printf*** é completar o número com zeros à direita, até que fique com seis casas decimais.

Para formatar de maneira diferente usar-se, junto com o ***%f*** uma especificação de quantas casas decimais se deseja que o número tenha. Especifica-se também o número total de caracteres do número a ser impresso.

Por exemplo: ***%6.3f*** especifica que se quer imprimir um **float** com **3 casas decimais** e com um **tamanho total** de **6**

caracteres no total.

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

void main()
{
   float NotaDaP1, NotaDaP2;
   float Media;

   clrscr();     // Limpa a tela

   NotaDaP1 = 6.6;  // Atribuição do Valores das médias
   NotaDaP2 = 8.2;

   Media = (NotaDaP1 + NotaDaP2) / 2.0;

   printf("Média Final : %6.3f", Media);
        // No momento da execução sinal %6.3f vai ser substituído
        // pelo valor da variável Media
        // Média Final :  7.400

   getch();      // espera que o usuário pressione uma tecla

}
```

Regras para impressão de um número real

- o número de casas decimais é sempre respeitado. Se for preciso, zeros serão acrescetados à direita do número
- o **tamanho total** significa o número de caracteres do número incluíndo o ponto decimal e um eventual sinal de menos (-), se for o caso;
- Se a soma do número de caracteres da **parte inteira**, mais o **ponto decimal**, mais a **parte fracionária**, mais um **eventual sinal de menos** *ainda for menor* do que o tamanho total especificado no formato, então, espaços em branco serão acrescentados à esquerda da parte real do número.
- Se a soma do número de caracteres da **parte inteira**, mais o **ponto decimal**, mais a **parte fracionária**, mais um **eventual sinal de menos** for maior do que o tamanho total especificado no formato, então, apenas o número de casas decimais é respeitado

Por exemplo:

```
#include <stdio.h>
#include <conio.h>
void main()
{
   float Numero;
   Numero = -2.5;
   clrscr();
   printf("1234567890\n");
   printf("%7f\n", Numero);
   printf("%7.0f\n", Numero);
   printf("%7.3f\n", Numero);
   printf("%8.3f\n", Numero);
   printf("%9.3f\n", Numero);
   printf("\n");
   printf("%8.4f\n", Numero);
   printf("%8.1f\n", Numero);
   printf("%6.12f\n", Numero);

   getch();
}

// Resultados

1234567890
-2.500000
     -2
 -2.500
  -2.500
```

```
   -2.500

 -2.5000
    -2.5
-2.500000000000
```

Alinhamento de números à DIREITA

Nos exemplos anteriores os números ficavam sempre alinhados a partir da esquerda.
Experimente colocar um sinal de menos logo depois do sinal de % e veja o que acontece.

```
          printf("%-7.3f\n", Numero);
```

Variáveis do Tipo String

Uma variável capaz de arrmazenar uma string deve ser declarada informando-se qual o número máximo de caracteres que ela poderá armazenar.

Exemplo:

```
       char Nome[30];  // isto define que a variável poderá armazenar uma
                       // string de até 29 caracteres.
```

Ao trabalharmos com strings deve-se incluir o arquivo de cabeçalho ***string.h***

Atribuição com strings

As atribuições de valores a strings **devem** ser feitas através da função **strcpy**

```
// Exemplo com strings

#include <stdio.h>
#include <conio.h>
#include <string.h>  // arquivo de cabeçalho para trabalhar com strings

void main()
{
   char Nome[30];  // declara uma string que poderá armazenar até 29 caracteres !!

   clsrscr();

   strcpy(Nome, "Jose da Silva");  // atribui "Jose da Silva" para a variável Nome
   printf("O funcionário %s foi tranferido", Nome);  // no lugar de %s aparecerá o
                                                     // conteúdo da variável Nome

   getch();
}
```

Operadores Aritméticos

-	sinal de menos (unário)	Maior precedência (avaliado antes)
*,/	multiplicação e divisão	
%	módulo (reto da divisão)	
+,-	soma e subtração	Menor precedência (avaliado depois)

Pode-se usar parênteses para alterar a precedência.

Exercício

Crie um progama que organize os dados em uma tabela conforme o exemplo a seguir. Os dados das células em amarelo deve ser informados através de atribuições dentro do programa.

Procure usar somente os %f para definir o formato e a posição dos dados.
Não tente preencher os espaços com brancos. Por exemplo, use

```
printf ("%10d, QuantidadeDeBananas);
```

ou invés de

```
printf("          %d", QuantidadeDeBananas);
```

Produto	Preço Unitário	Quantidade	Preço Total
Banana	R$ 2.50	2	R$ 5.00
Uva	R$ 6.50	6	R$ 39.00
Pessego	R$ 10.22	10	R$ 102.20
		Sub-Total	R$ 146.20
	Imposto (5%)		R$ 7.31
		Total	R$ 153.51