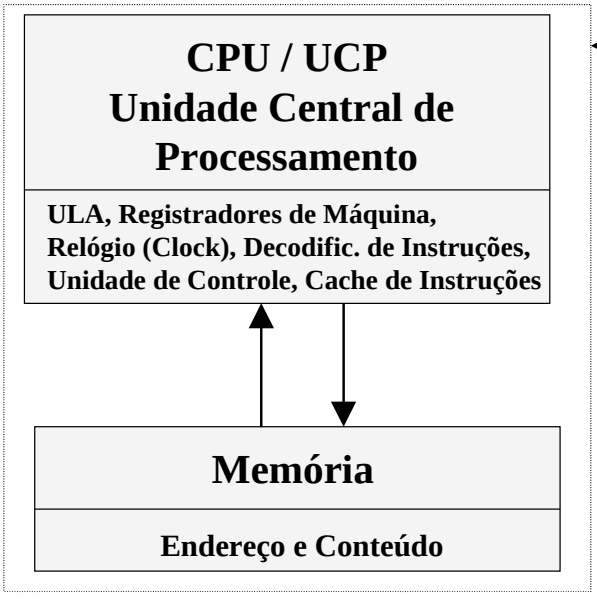
PIP/CA - ***Programa Interdisciplinar de Pós-Graduação em Computação Aplicada da UNISINOS***

- **Disciplina de Nivelamento - 2000/1:**

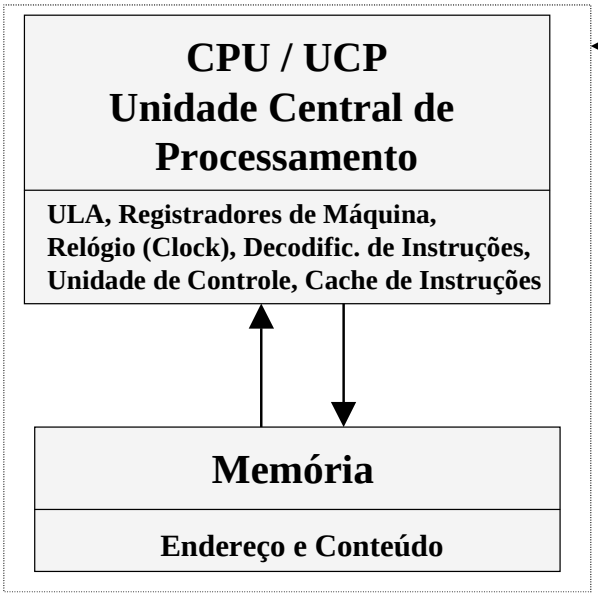
ALGORITMOS
&
ESTRUTURAS DE DADOS

- **Professor Responsável:**

Prof. Fernando Santos Osório
E-Mail: osorio@exatas.unisinos.br
Web: http://www.inf.unisinos.br/~osorio/



E/S
Periféricos:
- **Vídeo => *StdOut***
- **Teclado => *StdIn***
- **Impressora**
- **Disco**
- **CD-ROM ...**

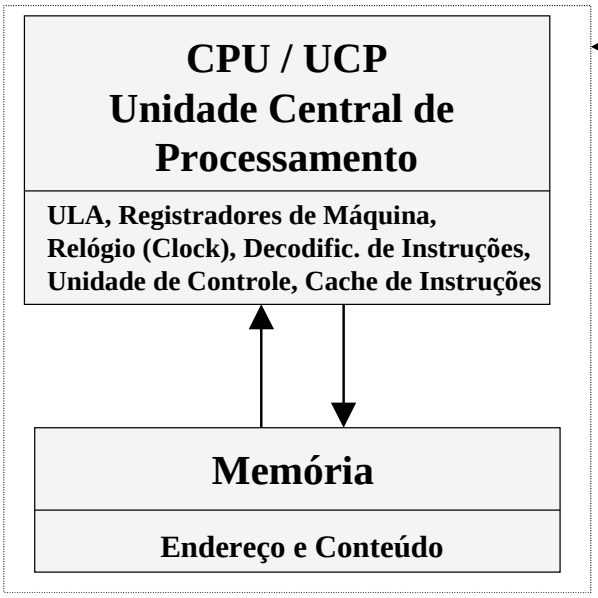


E/S

Periféricos:

- **Vídeo => *StdOut***
- **Teclado => *StdIn***
- **Impressora**
- **Disco**
- **CD-ROM ...**

***CPU*: Códigos de Máquina (Assembly - Mnemônicos)**
***Homem*: Linguagens de Alto Nível (Próximas da Linguagem Natural Humana)**



E/S

Periféricos:

- **Vídeo => *StdOut***
- **Teclado => *StdIn***
- **Impressora**
- **Disco**
- **CD-ROM ...**

***CPU*: Códigos de Máquina (Assembly - Mnemônicos)**
***Homem*: Linguagens de Alto Nível (Próximas da Linguagem Natural Humana)**

Programação:	**Alto Nível**	**Baixo Nível**
• Imperativa	Pascal	Assembly
• Determinística	"C" / C++	8080
• Estruturada	Java	80x86
• OOP (Objetos)	Cobol ...	680xx ...

Ferramentas de Software:

- **Sistema Operacional**
- **Compiladores - GCC**
- **Interpretadores - PERL**
- **Compilador/Interpretador: Java (JVM)**
- **Aplicativos: Word, Excel, Browsers, ...**

E/S

Periféricos:

- **Vídeo => *StdOut***
- **Teclado => *StdIn***
- **Impressora**
- **Disco**
- **CD-ROM ...**

***CPU*: Códigos de Máquina (Assembly - Mnemônicos)**
***Homem*: Linguagens de Alto Nível (Próximas da Linguagem Natural Humana)**

Programação:	Alto Nível	Baixo Nível
• Imperativa	Pascal	Assembly
• Determinística	"C" / C++	8080
• Estruturada	Java	80x86
• OOP (Objetos)	Cobol ...	680xx ...

Linguagem "C" :

- Criada por Kernighan e Ritchie
- Linguagem mais utilizada em ambientes acadêmicos, de pesquisa e de desenvolvimento de ferramentas básicas

PIP/CA - Adotaremos a linguagem "C" inicialmente como ferramenta para desenvolvimento de programas

***Motivos da escolha desta linguagem*:**

- **Portabilidade (GCC for Windows / GCC for Linux)**
- **Bem estruturada, gera código otimizado**
- **Flexibilidade, potencialidade ("ling. aberta"), C++**
- **Uso de *Software Livre* e das bibliotecas disponíveis**
- **Ambiente de desenvolvimento: IDE**
  Integrated Development Environment:
  *** Rhide for Windows - Editor, Compilador,Depurador (debug)**
  *** Linux: gcc, vi/xedit/emacs/pico, xxgdb**

Linguagem “C” : Exemplo de programa em “C”

DOS> type hello.c

```
#include <stdio.h>

main ( )
{
    printf (“\n”);
    printf (“Hello World! \n”);
    printf (“\n”);
}
```

DOS> gcc hello.c -o hello.exe -lm

LINUX> cat hello1.c

```
#include <stdio.h>

char nome [30];

main ( )
{
    printf (“Nome? “);
    scanf  (“%s”,nome);
    printf (“\n”);
    printf (“Hello %s!\n”,nome);
    printf (“\n”);
}
```

LINUX> gcc hello1.c -o hello1 -lm

Tipos de Dados em “C” : Tipos Básicos

Modificador	Tipo	Descrição
	CHAR	Caracter ASCII (valores de 0 a 255 / -128 a + 127 = 1 byte)
UNSIGNED	CHAR	Byte com valores positivos, sem sinal (0 a 255)
SIGNED	CHAR	Byte com valores negativos, com sinal (-128 a +127)
	CHAR[x]	String de no máximo ‘x’-1 componentes (caracteres)
	INT	Valor inteiro (usualmente de 2 ou de 4 bytes)
UNSIGNED	INT	Valor inteiro positivo
SIGNED	INT	Valor inteiro positivo ou negativo, com sinal
SHORT	INT	Valor inteiro com precisão inferior
LONG	INT	Valor inteiro com precisão superior
	INT[x]	Vetor de inteiros com ‘x’ componentes (valores inteiros)
	FLOAT	Valor de ponto flutuante (c/casas decimais)
UNSIGNED	FLOAT	Valor de ponto flutuante positivo
	DOUBLE	Valor de ponto flutuante com dupla precisão
LONG	DOUBLE	Valor de ponto flutuante com dupla precisão estendida
	ENUM	Enumeração de elementos. Exemplo: Seg, Ter, Qua, Qui, Sex
	VOID	Tipo de dados indefinido. Indica apenas o endereço da memória, sem no entanto especificar o tipo exato.

Modificadores: Signed, Unsigned, Short, Long, Static, Register

Tipos de Dados em "C" : Criação dos Dados

- Declarando valores constantes...

```
#define  MAXIMO   1000                  /* Início: Isto é um comentário...          Fim */
#define  VERSAO    1.0                  /* O #define é uma macro do pré-processador */
#define  SISOP     "LINUX"
const    double    valor_do_pi = 3.1415926;    /* Cria uma "variável constante"      */
```

- Declarando variáveis...

```
int idade;                          /* Variáveis podem ser declaradas antes de começar  */
double salario;                     /*  a parte que descreve as rotinas do programa     */
char sexo;                          /* Estas variáveis são chamadas de GLOBAIS          */

 main( )
{                                   /* Sempre após um "abre chaves" posso declarar vars. */
  int dia;                          /* Estas variáveis pertencem ao bloco definido entre o  */
  int mês;                          /* abre chaves '{' e o fecha chaves '}'                 */
  int ano;                          /* Estas variáveis são chamadas de LOCAIS do bloco   */
  ...
}
```

Observações importantes:
- **Nomes de variáveis devem começar por uma letra, seguidos de letras, dígitos ou '_';**
- **Maiúsculas e minúsculas são diferenciadas no nome das variáveis;**
- **Atenção para não declarar variáveis com nomes de palavras reservadas: for, while, int, ...**

Tipos de Dados em "C" : Atribuição de valores as variáveis

- Atribuindo valores na declaração:

```
float  cotacao_dolar = 1.87;
char  ativo = 'S';
```

- Atribuindo valores diretamente nas variáveis:

```
int  dia;                       dia = 21;
float salario;                  salario = 100.00;
double tabela[10];              tabela[0] = 0.001;
char nome[30];                  strcpy(nome,"Fulano");
```

- Expressões:

```
 int a,b;  double c;            c = a / b;      /*  Cuidado com os tipos de dados!! */
                                c = a + b / 3.0;   /* Cuidado com a precedência!! */
```

- Operador de conversão explícita de tipos de dados:

```
 int valint;                    valdoub = (double)valint;
 double valreal;                valint    = (int)valdoub;      /* Conversão explícita    */
                                valint    = valdoub;           /* Conversão implícita    */
                                valdoub  = valint/(double)outro_valint;   /* Forçada  */
```

Op.	Função	Exemplo "C"	Exemplo PASCAL
-	Menos unário	A = -B	A := -B
+	Mais unário	A = +B	A := +B
!	Negação Lógica	! FLAG	not FLAG
~	Bitwise NOT	A = ~B	A := not B
&	Endereço de	A = &B	A := ADDR(B)
*	Referência a ptr	A = *ptr	A := ptr^
sizeof	Tamanho de var	A = sizeof(b)	A := sizeof(b)
++	Incremento	++A ou A++	A := succ(A)
--	Decremento	--A ou A--	A := pred(A)
*	Multiplicação	A = B * C	A := B*C
/	Divisão inteira	A = B / C	A := B div C
/	Divisão real	A = B / C	A := B / C
%	Módulo da divisão	A = B % C	A := B mod C
+	Soma	A = B + C	A := B + C
-	Subtração	A = B - C	A := B - C
>>	Shift Right	A = B >> N	A := B shr N
<<	Shift Left	A = B << N	A := B shl N
>	Maior que	A > B	A > B
>=	Maior ou igual a	A >= B	A >= B
<	Menor que	A < B	A < B
<=	Menor ou igual a	A <= B	A <= B
==	Igual a	A == B	A = B
!=	Diferente de	A != B	A <> B
&	Bitwise AND	A = B & C	A := B and C
|	Bitwise OR	A = B | C	A := B or C
^	Bitwise XOR	A = B ^ C	A := B xor C
&&	Logical AND	flg1 && flg2	flg1 and flg2
||	Logical OR	flg1 || flg2	flg1 or flg2
=	Assinalamento	A = B	A := B
OP=	Assinalamento	A OP= B	A := A OP B

Tipos de Dados em "C" : Criando novos tipos e tipos compostos

- Comando TYPEDEF : Criando novos tipos

```
typedef   enum  { seg, ter, qua, qui, sex } dias_semana;
typedef   enum  { sab, dom } fim_de_semana;
typedef   short int     tipo_ano;

tipo_ano        ano_nascimento;         /* A variável ano_nascimento é do tipo "short int"    */
dias_semana     compromisso;            /* compromisso é uma variável do tipo dias_semana */
```

- Comando STRUCT: Criando tipos compostos (registros)

```
struct   data {
                int  dia;
                int  mes;
                int  ano;
              } ;

struct   data    data_nasc;

data_nasc.dia = 1;
data_nasc.mes = 1;
data_nasc.ano = 2000;
```

```
typedef   struct {
                    long int  nro_funcionario;
                    double    salario;
                    data      data_contratacao;
                  }   reg_funcionario;

reg_funcionario  diretor_dept_pessoal;

diretor_dept_pessoal.nro_funcionario = 1234567;
diretor_dept_pessoal.salario = 9999.99;
diretor_dept_pessoal.data_contratacao.ano = 1999;
```

Tipos de Dados em "C" : Criando novos tipos e tipos compostos

- Comando TYPEDEF : Criando novos tipos

```
typedef   enum  { seg, ter, qua, qui, sex } dias_semana;
typedef   enum  { sab, dom } fim_de_semana;
typedef   short int     tipo_ano;

tipo_ano        ano_nascimento;          /* A variável ano_nascimento é do tipo "short int"    */
dias_semana     compromisso;             /* compromisso é uma variável do tipo dias_semana */
```

- Comando STRUCT: Criando tipos compostos (registros)

```
struct  {
                int  dia;
                int  mes;
                int  ano;
        }  data;

data    data_nasc;

data_nasc.dia = 1;
data_nasc.mes = 1;
data_nasc.ano = 2000;
```

```
typedef   struct  {
                long int  nro_funcionario;
                double    salario;
                data      data_contratacao;
              }   reg_funcionario;

reg_funcionario  diretor_dept_pessoal;

diretor_dept_pessoal.nro_funcionario = 1234567;
diretor_dept_pessoal.salario = 9999.99;
diretor_dept_pessoal.data_contratacao.ano = 1999;
```

Tipos de Dados em "C" : Vetores

- Vetores numéricos:

```
int       Hora[24];          =>   Hora[0] . . Hora[23]  com valores do tipo "int"
double Notas[10];            =>   Notas[0] . . Notas[9]  com valores do tipo "double"
Notas[0] = 10.0;
```

N[0]	N[1]	N[2]	N[3]	N[4]	N[5]	N[6]	N[7]	N[8]	N[9]

- Vetores de caracteres:

```
char    Letras[26];          =>   Letras[0] . . Letras[25] com valores do tip "char"
Letras[0] = 'a';  Letras[25] = 'z';

char    Nome[10];                 => Nome[0] . . Nome[9] onde uma posição é reservada para a
                                     marca de fim da string de nome!   (Marca = '\0')
strcpy(Nome,"123456789");    => O Nome não deve ter mais de 9 caracteres, pois o décimo é o '\0'
                                  Strings são manipuladas através de rotinas especiais:
                                  strcpy, strlen, strcmp, sprintf, sscanf, ...     #include <string.h>
```

N[0]	N[1]	N[2]	N[3]	N[4]	N[5]	N[6]	N[7]	N[8]	N[9]
F	U	L	A	N	O	\0	?	?	?

Tipos de Dados em "C" : Vetores bi-dimensionais

- Vetores numéricos bi-dimensionais:

int Matriz [3][10];

Matriz[0][0] = 1; ... Matriz [2][9] = 30;

M[0][0]	M[0][1]	M[0][2]	M[0][3]	M[0][4]	M[0][5]	M[0][6]	M[0][7]	M[0][8]	M[0][9]
M[1][0]	M[1][1]	M[1][2]	M[1][3]	M[1][4]	M[1][5]	M[1][6]	M[1][7]	M[1][8]	M[1][9]
M[2][0]	M[2][1]	M[2][2]	M[2][3]	M[2][4]	M[2][5]	M[2][6]	M[2][7]	M[2][8]	M[2][9]

- Inicialização de vetores:

int num [5] = { 1, 2, 3, 4, 5 };

char vogais[5] = { 'a', 'e', 'i', 'o', 'u' };

```
double matriz [3][2] = {  { 0,0 }, { 0,1 },
                          { 1,0 }, { 1,1 },
                          { 2,0 }, { 2,1 }  };
```

Tipos de Dados em "C" : Constantes do tipo caracter

```
const <tipo_var> <nome_const> =  <valor>
# DEFINE <nome_const> <valor>   <==   Macro do pré-processador

Exemplos de constantes :

255     ->  Decimal       *  constantes Inteiras
0777    ->  Octal         *  começando com 0 -> Octal
0XFF    ->  Hexadecimal   *  começando com 0x -> Hexa
nnnL    ->  Long          *  L,U -> forçam uso do tipo
nnnU    ->  Unsigned      *  podem ser usados juntos
nnnF    ->  Float         *  força armazenamento como float
'c'     ->  Caracter (seu valor é o ascii de 'c' - tipo char)
'CC'    ->  Caracter ( 2 caracteres - tipo int ) (TC)
                \XXX    ->  Octal XX (expl.: '\014' é o form feed).
                            São esperados 3 dígitos após o "\" !
\n      ->  New Line        LF  0x0A   * Sequências de escape
\b      ->  Backspace       BS  0x08
\r      ->  Carriage Return   CR  0x0D
\f      ->  Form Feed       FF  0x0C
\t      ->  Tab Horiz       HT  0x09
\v      ->  Tab Vertical    VT  0x0B
\0      ->  Null            0x00
\\      ->  Backslash       \   0x5C
\'      ->  Single quote     '   0x2C
\"      ->  Double quote     "   0x22
\?      ->  Question Mark    ?   0x3F
\a      ->  Bell            BEL 0x07   (ANSI)
\xNN    ->  Repr. hexa       0xNN   (ANSI)
```

Comandos da Linguagem "C" :

IF

if (<*expressão*>)
** <*comando*>;**
else
** <*comando*>;**

Expressão:
- Expressão lógica, relacional, aritmética

Comando:
- Comando simples ou bloco de comandos
- Bloco de comandos: { ... }
{ comando;comando; ... } ~ comando;

```
if  ( salario > 100.00)
    printf ("Salário maior que R$100,00\n");

if ( salario == 0.00)
   printf ("Este já foi demitido faz tempo...\n");
else
   printf ("Este ainda está sendo pago...\n");
```

CUIDADO: ***if (a == b)*** ***NÃO É O MESMO QUE*** ***if (a = b) !!***

Comandos da Linguagem "C" :

SWITCH-CASE

```
switch ( <expressão> )    /* Com resultado do tipo int ou char */
{
   case   <valor1> :  <comando> ;
                          break;
   case   <valor2> :  <comando> ;
                          break;
   ...
   default :  <comando>;
}
```

>> O comando switch é um comando que permite estruturar melhor um conjunto de IF's aninhados.

Comandos da Linguagem "C" :

FOR

for (<*expr_inicial*>; <*condição_de_parada*>; <*alteração_var_controle*>)
<*comando*>;

```
for ( contador=0; contador < nro_vezes; contador++ )
     printf("Contando... %d\n", contador);

for (contador=10; contador != 0; contador-- )
{
     printf("Contagem regressiva...\n");
     printf("Falta: %d \n", contador);
 }
```

CUIDADO: `for ( a=1; a <= 10; a++) ;`
`for ( ; ; )  /* Loop infinito */`

Comandos da Linguagem "C" :

WHILE

while (<*expr_inicial*>)
<*comando*>;

```
contador = 0;
while ( contador < nro_vezes )
     printf("Contando... %d \n", contador++);

contador = 10;
printf("Contagem regressiva...\n");
while ( contador != 0 )
{
   printf("Falta: %d \n", contador);
   contador--;
 }
```

CUIDADO: `while (contador < fim) ; contador++;`

Comandos da Linguagem "C" :

DO-WHILE

do <*comando*>
while (<*expr_inicial*>) ;

```
contador = 0;
do printf ("Contando... %d \n", contador++)
while ( contador < nro_vezes );

contador = 10;
printf("Contagem regressiva...\n");
do  {
        printf ("Falta: %d \n" , contador);
        contador--;
    }
 while ( contador != 0 );
```

Comandos da Linguagem "C" :

BREAK

break; => Força o término da execução de um loop

```
for ( ; ; )
{
    printf ("Resposta: ");
    scanf  ("%d",&valor);
    if (valor == VALOR_CORRETO)  break;
}

contador = 0;
while ( 0 == 0 )
{
        printf ("Contando... %d \n", contador++)
        if ( contador > MAXIMO)  break;
}
```

Comandos da Linguagem "C" :

CONTINUE

continue; => Força o recomeço da execução de um loop

```
for ( ; ; )
{
      printf ("Resposta: ");
      scanf  ("%d",&valor);
      if (valor != VALOR_CORRETO)
        continue;
      printf ("Bravo !\n");
      break;
}
```

Comandos da Linguagem "C" :

EXIT

exit; => Força o término da execução do programa!

```
printf ("Resposta: ");
scanf  ("%d",&valor);
if (valor > LIMITE_MAXIMO)
{
    printf ("Erro: valor inválido. Valor superior ao limite máximo!\n");
    exit(0);
}
```

>> O valor entre parênteses é retornado ao sistema operacional.

Comandos da Linguagem "C" :

RETURN

return; => Retorna para a função que chamou esta sub-rotina.
Permite o retorno de um valor.

```
#include <stdio.h>

int verifica_idade ( idade )
int idade;
{
    if (idade >= 18)
       return (TRUE);
    else
       return (FALSE);
}
```

```
main ( )
{
    int  anos;

    printf("Idade: ");
    scanf ("%d",&anos);
    if ( verifica_idade(anos) )
       printf("Maior\n");
    else
       printf("Menor\n");
}
```

Estrutura de programas na Linguagem "C" :

```
/* Exemplo de um programa em "C" */

#include <stdio.h>              /* Definições externas */
#define   MAXIMO  10            /* Valor constante     */

int Matriz [MAXIMO];            /* Variável Global     */

/* Rotinas do Programa */

void main ( )                   /* Nome e parâmetros */
{
   int  aux;                    /* Variável Local      */

   printf("Entre com %d valores:\n",MAXIMO);
   for ( aux = 0; aux < MAXIMO; aux++)
       scanf ("%d", &(Matriz[aux]) );
   printf("Valores fornecidos:\n");
   for ( aux = 0; aux < MAXIMO; aux++)
       printf("Matriz[%d] = %d \n", aux, Matriz[aux]);
}
```

Funções da Linguagem "C" : Entrada e Saída

• **PRINTF - Escrita de dados na tela (stdout)**

printf (<string_de_controle>, <variável>, <variável>, ...);

Exemplos:

```
printf ("Escrevendo apenas um texto na tela.");
printf ("No final pula para a linha seguinte.\n");
printf ("Valor inteiro = %d",variavel_int);
printf ("Valor float   = %f",variavel_float);
printf ("Valor double = %lf",variavel_double);
printf ("Caracter      = %c",variavel_letra);
printf ("String (vetor de caracteres terminado por NULL) = %s",nome);
printf ("Números: %d, %lf, %d\n", 2, 3.1415, variavel_int);
printf ("Valor com apenas 3 casas após a vírgula: %.3lf",variavel_double);
printf ("Hoje é %d/%d/%d\n",dia, mes, ano);
```

Arquivos: fprintf (<arquivo>, <string_de_controle>, <variável>, ...);
Strings : sprintf (<variavel>, <string_de_controle>, <variável>, ...);

Funções da Linguagem "C" : Entrada e Saída

•**SCANF - Leitura de dados do teclado (stdin)**

scanf (<string_de_controle>, [&]<variável>, [&]<variável>, ...);

Exemplos:

```
scanf ("%d", &variavel_int);
scanf ("%f", &variavel_float);
scanf ("%lf", &variavel_double);
scanf ("%c", &variavel_char);                 /* Humm...              */
scanf ("%s", vetor_de_caracteres);            /* NÃO tem o '&' !!! */
scanf ("%d %lf %d", &vint, &vdouble, &vint);
scanf ("%d %d %d", &dia, &mes, &ano);
```

Arquivos: fscanf (<arquivo>, <string_de_controle>, <variável>, ...);
Strings : sscanf (<variavel>, <string_de_controle>, <variável>, ...);
Outras funções de leitura do teclado:

```
vchar   = getch ( );        /* Lê um caracter sem esperar o Enter     */
vstring = gets ( );         /* Lê uma string terminada por um Enter */
```