Algoritmos

Linguagem de 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 algumas 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, Carteiro, PesoDoOnibus,
  Usuario_1, CorDoLivro, RaioDaTerra

Variáveis

Uma variável é uma posição 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 memória.

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 bytes	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 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.

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");
}
```

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 Centro “CENTRO FEDERAL DE EDUCAÇÃO TECNOLÕGICA" // 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", Centro); // Imprime o nome representado pela constante
   getchar (); // espera que o usuário pressione uma tecla
}
```

Algoritmos

Linguagem de programação C

Parte 2

Definição de constantes

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


#define PI 3.1416
#define VERSAO "2.02"


main ()
{
        printf ("Programa versao %s \n \n",VERSAO);
        getchar();


        printf ("O numero pi vale: %f",PI);
        getch();
}
```

Impressão de inteiros com printf

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

void main ()
{
     int Contador;
     int NroDeFilhos;

    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

```
#include <stdio.h>
#include <conio.h>   // necessário para as funções getch

main ()
{
   int NroDeAndares;
   int AlturaPorAndar;


   NroDeAndares = 10;
   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

```
#include <stdio.h>
#include <conio.h>   // necessário para as funções getch

main()
{
  float NotaP1, NotaP2;
  float Media;



  NotaP1 = 3.0;  // Atribuição do Valores das médias
  NotaP2 = 5.5;

  Media = (NotaP1 + NotaP2) / 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 : 4.250000
  getch();      // espera que o usuário pressione uma tecla
}
```

Formato de Impressão dos Números Reais

No exemplo acima o resultado da média (4.25) foi impresso com 6 casas decimais (4.250000).

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 usa-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>

main()
{
  float NotaP1, NotaP2;
  float Media;

  // Limpa a tela

  NotaP1 = 9.0;  // Atribuição do Valores das médias
  NotaP2 = 5.0;

  Media = (NotaP1 + NotaP2) / 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 :  7.000

  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

```
#include <stdio.h>
#include <conio.h>
main()
{
   float Numero;

   Numero = -2.5;

   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();
}
```

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

main()
{
    int i;
    float f;


  printf( "Entre com um numero inteiro: " );
  scanf( "%d", &i );

  printf( "Entre com um número ponto flutuante: " );
  scanf( "%f", &f );

  printf( "Você entrou primeiro com o número %d e depois %f\n", i, f );
  getch();

}
```

Comandos de controle de fluxo no C

Os comandos de controle de fluxo são a essência de qualquer linguagem de programação, porque governam o fluxo da execução do programa.

Os comandos de controle de fluxo são divididos em três categorias. A primeira consiste nas instruções condicionais if. A segunda são os comandos de controle de loop while, for e do-while.

Observação: Lembre-se de que um comando pode consistir em um dos seguintes itens: um comando simples, um bloco de comandos.

Comando If

A forma geral do comando if é:

If (condição) comando;

else comando;

A cláusula else é opcional. Se condição avaliar em veradeiro, o computador executará o comando ou bloco que forma o objetivo de if; de outro modo, se a cláusula else existir, o computador executará o comando ou bloco que é o objetivo de else;

Lembre-se: de que apenas o código associado a if ou o código associado a else será executado - nunca os dois.

Comando If

```
#include <stdio.h>
#include <conio.h>
main()
{
  if (getche()==‘t')
   {
    printf ("\ A tecla pressionada foi a tecla t.\n");
    printf("Pressione qualquer tecla para terminar.");
    getche();
   }
}
```

Comando If

Exemplo: Considere o seguinte programa, que é uma versão simples do jogo “adivinhe o número mágico”. O programa imprime a mensagem “Certo” quando você adivinha o número mágico e usa a função de biblioteca **rand( )** para gerá-lo. A função **rand( )** devolve um inteiro aleatório na escala de 0 a 32767:

Comando If

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

main()
{
   int magico, palpite;

   srand((unsigned) time(NULL));
   magico = rand()%10; //gera o numero mágico

   printf("\nAdivinhe o numero magico de 0-9: ");
   scanf("%d", &palpite);

   if( palpite == magico )
                printf("Certo! ");
   else
                printf("Errado! ");
   printf("O numero magico eh: %d\n", magico);
   getch();

}
```

Comando If aninhados

Um dos aspectos que mais causam confusão em relação aos comandos if em qualquer linguagem é o if aninhado. Um if aninhado é um comando if que é objeto de um if ou de um else. A razão para tanta confusão é que você pode ter dificuldades em dizer se o else está relacionado com qual if. Análise o exemplo abaixo:

```
If (x)
   if (y) printf (“1”);
   else printf (“2”);
```

A qual if o else se refere?

Comando If aninhados

Para nossa sorte o C tem uma regra simples para esse problema.

• Em C, o else está sempre ligado ao if mais próximo que ainda não tiver um comando else associado a ele.

• Tanto o if como o else devem estar dentro do mesmo bloco de código. No caso anterior, o else está associado ao comando if(y).

• Para fazer que ele se associe a if(x), você deve usar chaves para anular a associação normal, dá seguinte forma.

```
If (x)
{
  if (y)  printf("1");
}
else printf("2");
```

• O else agora está associado a if (x), porque ele não faz mais parte do bloco do código if (y)

Comando If aninhados

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

main()
{
  int magico, palpite;

  srand((unsigned) time(NULL));
  magico = rand()%10; //gera o numero mágico

  printf("\nAdivinhe o numero magico de 0-9: ");
  scanf("%d", &palpite);

   if(palpite == magico) {
                printf("Certo! ");
                printf("%d é o número mágico \n", magico);
   }
   else {
      printf("Errado! ");
      if (palpite>magico)
        printf("Muito alto \n");
      else printf ("Muito baixo\n");

    }
    getch();


}
```

Comando If else-if

Uma construção comum em programação é a escada if else-if

```
if (condição)
  comando;
else if (condição);
  comando;
else if (condição)
   comando;
     *
     *
     *
 else

comando;
```

Comando If else-if

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

main()
{
  int magico, palpite;

  srand((unsigned) time(NULL));
  magico = rand()%10; //gera o numero mágico

  printf("\nAdivinhe o numero magico de 0-9: ");
  scanf("%d", &palpite);

   if(palpite == magico) {
                printf("Certo! ");
                printf("%d é o número mágico \n",
magico);
    }
    else if (palpite>magico)
          printf("Errado --Muito alto \n");
       else
       printf ("Muito baixo\n");

    getch();
}
```

Comando Switch

• Embora a escada if else-if possa executar testes de várias maneiras, ela não é de maneira nenhuma elegante.

• O código pode ficar difícil de ser seguido e confundir até mesmo o seu autor no momento futuro.

• Por esse motivo, o C possui um comando íntrinseco de vários desvios chamado switch.

• O computador testa uma variável sucessivamente contra uma lista de constantes inteiras ou de caracteres.

• Depois de encontrar uma coincidência, o computador executa o comando ou blocos de comandos que estejam associados àquela constante.

Comando Switch

- A forma geral do comando switch é:

```
switch (variável) {
  case constante 1:
      sequência de comandos
      break;
  case constante 2:
      sequência de comandos
      break;
   case constante 3:
      sequência de comandos
      break;
    *
    *
    *
      default:
       sequência de comandos

}
```

Comando Switch

- O computador executa o comando default se não encontrar nenhuma coincidência.

- O comando default é opcional. Se ele não estiver presente não ocorrerá nenhuma ação no caso de todas as comparações falharem.

- Quando ele encontra uma coincidência, o computador executa os comandos associados àquele case até encontrar um break.

Comando Switch

Observações importantes:

• Switch difere do if, pois o primeiro só pode testar igualdade e a expressão condicional if pode ser de qualquer tipo.

• Não pode haver duas constantes case com valores iguais no mesmo switch.

• O comando switch normalmente é utilizado para processar comandos do teclado como, por exemplo, a escolha de um menu.

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

main()
{
  char ch;
  printf("1. Verificar ortografia\n");
  printf("2. Corrigir ortografia\n");
  printf("3. Exibir erros de ortografia\n");
  printf("4. Pressione qualquer outra tecla para sair\n\n");
  printf("      Digite sua opção:");

  ch=getchar();// Ler seleção do teclado

  switch(ch){
    case '1':
        printf("verificar");
        break;

    case '2':
        printf("corrigir");

        break;
    case '3':
        printf("exibir");
        break;

    default:
     printf(" Nenhuma opção foi escolhida");

  }
  getche();
}
```

Loops

Os loops permitem que o computador repita um conjunto de instruções até que alcance uma certa condição. O C aceita os seguintes tipos de loop:

- for
- while
- do-while

Conceitos básicos do loop for

A forma geral comando for é:

for (inicialização; condição;incremento) comando;

Como simples exemplo, o programa a seguir imprime na tela os números de 1 a 100:

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

main()
{
int x;
for(x=1;x<=100;x++) printf("%d\n", x);
getch();
}
```

Conceitos básicos do loop for

Refaça o exemplo anterior, entretanto agora os limites inferior e superior do laço devem ser fornecidos pelo usuário.

Dica: use o camando scanf para leitura

Conceitos básicos do loop for

O loop for não precisa sempre fluir para a frente.

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

main()
{
int x;
for(x=100;x>0;x--) printf("%d\n", x);
getch();
}
```

Conceitos básicos do loop for

Refaça o exemplo anterior, entretanto agora os limites inferior e superior do laço devem ser fornecidos pelo usuário.

Dica: use o camando scanf para leitura

Conceitos básicos do loop for

Refaça o exemplo anterior, entretanto agora os limites inferior e superior do laço devem ser fornecidos pelo usuário. Levando em conta que normalmente o usuário que utiliza o programa pode digitar os valores de forma invertida, utilize uma estrutura if para testar se ele digitou corretamente (a>b, por exemplo), em caso contrário, deixar uma mensagem (else).

Dica: use o camando scanf para leitura

Conceitos básicos do loop for

Você também não está somente restrito ao incremento e decremento da variável De controle.

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

main()
{
int x;
for(x=0;x<=100;x=x+5) printf("%d\n", x);
getch();
}
```

Conceitos básicos do loop for

Refaça o exemplo anterior, entretanto agora os limites inferior e superior do laço devem ser fornecidos pelo usuário e também o incremento para o laço. Levando em conta que normalmente o usuário que utiliza o programa pode digitar os valores de forma invertida, utilize uma estrutura if para testar se ele digitou corretamente (a>b, por exemplo), em caso contrário, deixar uma mensagem (else).

Dica: use o camando scanf para leitura

Conceitos básicos do loop for

Você também pode usar o comando for para repetir vários comandos

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

main()
{
int i;
  for(i=0;i<100;i++){
     printf("Isto é i: %d ", i);
     printf("é o quadrado de i: %d\n", i*i);

   }
getch();
}
```

Comando While

- O segundo Loop disponível em C é o Loop while. Sua forma geral é:

while(condição) comando

- comando, pode ser um comando simples ou um bloco de comandos.
- A condição pode ser qualquer expressão válida.

Comando while

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

main()
{
  char ch;
  ch='0'; // Iniciando ch com um caractere
  while(ch!='A') ch=getche();
}
```

Comando while

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

main()
{
  int x;
  x=1;

  while(x<=20000){
    printf("Calma ainda estou calculando!!!\n");
    x=x+1;
  }
  getch();
}
```

Comando do/while

Ao contrário do loop for e do loop while, que testam a condição no começo do loop, o loop do/while verifica a condição no final.

- Essa condição faz que o loop do/while seja executado pelo menos uma vez.

- A forma geral do loop do/while.

```
do{
    comando;

    }while (condição);
```

Comando do/while

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

main()
{
  int num;

  do{
      printf("Digite um numero, mas nao esqueça qual e o numero da condicao\n");
      scanf("%d",&num);

    }while(num<100);
}
```

Exercícios

1 – Faça um programa que receba quatro números inteiros, calcule e mostre a média.

2 – Faça um programa que receba três notas e seus respectivos pesos, calcule e mostre a média ponderada.

3- Faça um programa que receba o salário de um funcionário, calcule e mostre o novo salário sabendo que este recebeu um aumento de 25%.

4- Faça um programa que leia um conjunto de 10 cartões de uma turma, contendo cada um, a altura e o sexo de uma pessoa, calcule e escreva.

i) A maior e a menor altura.
j) A média de altura das mulheres
k) A média de altura da turma

Vetores em C

Vetores

A forma geral de declaração de uma matriz unidimensional é:

tipo nome_da_variavel[tamanho]

O tipo determina o tipo de cada elemento da matriz.

O tamanho define quantos elementos a matriz conterá.

Exemplo:

int mostra[10]

Declaração de uma matriz do tipo inteiro chamada mostra e que tem tamanho de 10 elementos.

Observação:

Em C, todas as matrizes usam um zero como índice do primeiro elemento.

Portanto no exemplo anterior é declarado um vetor com 10 elementos

amostra[0]...amostra[9]

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
main()
{
   int x[10];
   int t;
   for(t=0;t<10;++t){
    x[t]=t;
    printf("o elemento %d do vetor[%d] e %d\n",t,t,t);
    }
   getche();
}
```

Cálculo da média de uma lista de números

```
main()
{
   int amostra[10],i,med;

   for(i=0;i<10;i++)
   {
      printf("Digite o numero %d\n",i);
      scanf("%d",&amostra[i]);
   }
   med=0;
   //soma dos números
   for(i=0;i<10;i++)
   {
   med=med+amostra[i];
   }
   printf("A media dos numeros e: %d\n",med/10 );
   getche();
}
```

A linguagem C não realiza verificação de limites em matrizes.

Por isso nada impede que você vá além do fim de uma matriz.

Se você transpuser o fim de uma matriz durante uma operação de atribuição, então você atribuirá valores a dados de outras variáveis.

```
main()
{
    int erro[10],i;

    for(i=0;i<30;i++) erro[i]=1;
   getche();
}
```

O vetor é uma matriz composta unidimensional (variável composta homogênea) do mesmo tipo

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

main()
{
    char cha[7];
    int i;

    for (i=0;i<7;i++)
    {
     cha[i]='A'+i;
          printf("A posicao %d tem o conteudo: %c\n",i,cha[i]);
    }
  getch();

}
```

Strings

Em C, uma string consiste em uma matriz de caracteres terminadas em zero. Um zero é
Especificado como ‘/0’.

Por essa razão você deve declarar, as matrizes de caracteres como sendo um caractere maior que a maior que a maior string que você quer que elas contenham.

Char str[11];

- Ainda que o C não tenha o tipo de dado string, ainda assim ele permite constantes de string, ainda assim ele permite constantes de String.

- Lembre-se que uma constante de string é uma lista de caracteres que aparece entre aspas.

Exemplo: “Bom dia” “Isto é um exemplo”

- O zero é acrescentado automaticamente no final da constante de String pelo compilador

Lendo uma string pelo teclado

A melhor forma de inserir uma string através do teclado é usar a função de biblioteca gets().

A forma geral é:

gets(nome_da_matriz)

Para ler uma string, você deve chamar gets() com o nome da matriz, sem qualquer índice, como argumento.

De acordo com o retorno de gets(), a matriz conterá a string introduzida pelo teclado.

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

main()
{
    char palavra[100] ;
    printf("Digite a frase ou palavra:");
    gets(palavra);
    printf("%s",palavra);
    getche();

}
```

Matriz

O C permite o uso de matrizes multidimensionais.

A forma mais comum da matriz multidimensional é a bidimensional.

Para declarar uma matriz bidimensional de tamanho 10, 20, você deve escrever

```
int bidimensional[10][20];
```

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

main()
{
    int t,i, num[3][4];

    for(t=0;t<3;t++)
    {
      for (i=0;i<4;i++)
      {
        num[t][i]=(t*4)+i+2;
        printf("%d\n",num[t][i]);
      }
    }
    getche();

}
```