Estruturas de Controle de Fluxo

1. Instrução condicional

```
if (<condição>)
   <instrução>
```

Estruturas de Controle de Fluxo

1. Instrução condicional (continuação)

```
if (<condição>)
   <instrução1>
else
   <instrução2>
```

Estruturas de Controle de Fluxo

```
if (<condição>)
{
   <instrução1a>
   .
   .
   .
   <instruçãonb>
}
else
{
   <instrução1b>
   .
   .
   .
   <instruçãomb>
}
```

Estruturas de Controle de Fluxo

Exemplo – O programa a seguir recebe como entrada dois números inteiros e gera como saída o resultado da divisão entre eles.

```
#include <stdio.h>
main()
{
        int n1, n2;
        float res;
        printf (“Digite o dividendo inteiro: ”);
        scanf (“%d”, &n1);
        printf (“Digite o divisor inteiro: ”);
        scanf (“%d”, &n2);
        if (n2==0)
          printf (“Impossivel dividir!”);
        else
        {
          res = n1 / n2;                                    /*apresentará o*/
          printf (“Resultado da divisao: %.2f”, res);/*quociente da divisão*/
        }                                                   /*inteira*/
}
```

Estruturas de Controle de Fluxo

Exemplo – O programa a seguir recebe como entrada dois números inteiros e gera como saída o resultado da divisão entre eles.

```
#include <stdio.h>
main()
{
        int n1, n2;
        float res;
        printf (“Digite o dividendo inteiro: ”);
        scanf (“%d”, &n1);
        printf (“Digite o divisor inteiro: ”);
        scanf (“%d”, &n2);
        if (!n2)
          printf (“Impossível dividir!”);
        else
        {
          res = n1 / (float)n2;
          printf (“Resultado da divisão: %.2f”, res);
        }
}
```

Estruturas de Controle de Fluxo

3. Laços de repetição

```
while (<condição>)
{
    <instrução1>
    .
    .
    .
    <instruçãon>
}
```

Estruturas de Controle de Fluxo

3. Laços de repetição (continuação)

Exemplo:

Dada uma reta ax+by+c=0 e cinco pontos, faça um programa para calcular, para cada ponto, o seguinte: se o ponto estiver no primeiro quadrante calcule e informe a distância do ponto a reta caso contrário escreva uma mensagem informando que o ponto não pertence ao primeiro quadrante.

UNIVASF

Estruturas de Controle de Fluxo

```
#include <stdio.h>
#include <math.h>
main()
{
   float a,b,c,x,y;
   int contador=1;
   printf ("Equação da reta: ax+by+c=0\n");
   printf ("Coeficiente a da reta = ");
   scanf ("%f",&a);
   printf ("\nCoeficiente b da reta = ");
   scanf ("%f",&b);
   printf ("\nCoeficiente c da reta = ");
   scanf ("%f",&c);
```

```
    while (contador<=5)
    {
       printf ("\nCoordenadas do ponto %d:\n", contador);
       printf ("\nCoordenada x do ponto = ");
       scanf ("%f",&x);
       printf ("\nCoordenada y do ponto = ");
       scanf ("%f",&y);
       if (x>=0.0 && y>=0)
          printf ("\nA distancia do ponto a reta eh: %f",
          fabs(a*x+b*y+c)/(float)sqrt(pow(a,2)+pow(b,2)));
       else
          printf ("\nO ponto nao esta no primeiro quadrante!");
       contador++;
    }
}
```

Estruturas de Controle de Fluxo

3. Laços de repetição (continuação)

```
do
{
    <instrução1>
    .
    .
    .
    <instruçãon>
}
while (<condição>);
```

Estruturas de Controle de Fluxo

3. Laços de repetição (continuação)

Exemplo:

Dada uma reta ax+by+c=0 e cinco pontos, faça um programa para calcular, para cada ponto, o seguinte: se o ponto estiver no primeiro quadrante calcule e informe a distância do ponto a reta caso contrário escreva uma mensagem informando que o ponto não pertence ao primeiro quadrante.

Estruturas de Controle de Fluxo

```
#include <stdio.h>
#include <math.h>
main()
{
   float a,b,c,x,y;
   int contador=1;
   printf ("Equação da reta: ax+by+c=0\n");
   printf ("Coeficiente a da reta = ");
   scanf ("%f",&a);
   printf ("\nCoeficiente b da reta = ");
   scanf ("%f",&b);
   printf ("\nCoeficiente c da reta = ");
   scanf ("%f",&c);
```

```
    do
    {
       printf ("\nCoordenadas do ponto %d:\n", contador);
       printf ("\nCoordenada x do ponto = ");
       scanf ("%f",&x);
       printf ("\nCoordenada y do ponto = ");
       scanf ("%f",&y);
       if (x>=0.0 && y>=0)
          printf ("\nA distancia do ponto a reta eh: %f",
          fabs(a*x+b*y+c)/(float)sqrt(pow(a,2)+pow(b,2)));
       else
          printf ("\nO ponto nao esta no primeiro quadrante!");
       contador++;
    } while (contador<=5);
 }
```

Estruturas de Controle de Fluxo

3. Laços de repetição

```
for (<instrução1>;<condição>;<instrução2>)
{
    <instrução3>
    .
    .
    .
}
```

Obs.: A condição é uma expressão lógica.

Estruturas de Controle de Fluxo

3. Laços de repetição (*for*)

1ª executar a instrução1

2ª avaliar a condição, se verdadeira executar 3ª, se falso sair do laço

3ª executar a instrução3

4ª executar a instrução2

5ª vá para o 2ª passo

Estruturas de Controle de Fluxo

3. Laços de repetição (observações)

```
for (i=0,j=0;i<10&&j<20;i++,j+=2)
{
    .
    .
    .
}
for (i=0;i<1000;i++);
printf ("\nEscreve qualquer coisa!\n");
/*O printf anterior não está contido no for!*/
```

3. Laços de repetição (continuação)

Exemplo:

Dada uma reta ax+by+c=0 e cinco pontos, faça um programa para calcular, para cada ponto, o seguinte: se o ponto estiver no primeiro quadrante calcule e informe a distância do ponto a reta caso contrário escreva uma mensagem informando que o ponto não pertence ao primeiro quadrante.

Estruturas de Controle de Fluxo

```
#include <stdio.h>
#include <math.h>
main()
{
   float a,b,c,x,y;
   int contador;
   printf ("Equação da reta: ax+by+c=0\n");
   printf ("Coeficiente a da reta = ");
   scanf ("%f",&a);
   printf ("\nCoeficiente b da reta = ");
   scanf ("%f",&b);
   printf ("\nCoeficiente c da reta = ");
   scanf ("%f",&c);
```

```
    for (contador=5;contador;contador--)
    {
       printf ("\nCoordenadas do ponto %d:\n",6-contador);
       printf ("\nCoordenada x do ponto = ");
       scanf ("%f",&x);
       printf ("\nCoordenada y do ponto = ");
       scanf ("%f",&y);
       if (x>=0.0 && y>=0)
          printf ("\nA distancia do ponto a reta eh: %f",
          fabs(a*x+b*y+c)/(float)sqrt(pow(a,2)+pow(b,2)));
       else
          printf ("\nO ponto nao esta no primeiro quadrante!");
    }
 }
```

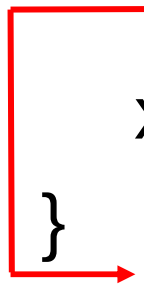
Estruturas de Controle de Fluxo

4. Comando *break*

Normalmente é utilizado em laços em que uma condição especial pode provocar uma terminação imediata.

Exemplo:

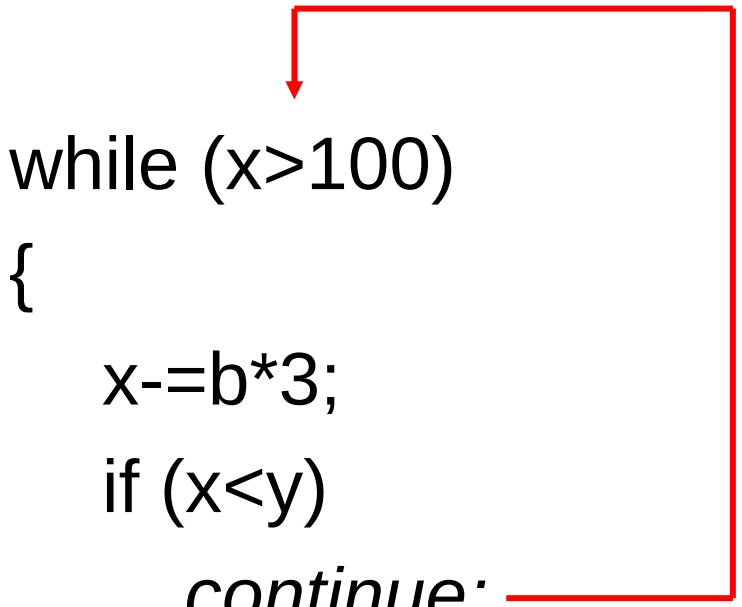
```
while (x>100)
{
    x-=b*3;
    if (x<100)
        break;
    x-=y*3;
}
```



Estruturas de Controle de Fluxo

5. Comando *continue*

Exemplo:



```
while (x>100)
{
    x-=b*3;
    if (x<y)
        continue;
    x-=y*3;
}
```

Estruturas de Controle de Fluxo

6. Seleção múltipla

```
switch (<variável_escalar>)
{
    case <constante1> : <instrução1>
                        break;
    case <constante2> : <instrução2>
                        break;
    .
    .
    .
    case <constantenn> : <instruçãon>
                        break;
    default:                <instruçãon+1>
}
```

```
/* Exemplo do switch */
#include <stdio.h> main()
{
   int a,b,c;
   printf("\nEntre com o valor de a : ");
   scanf("%d",&a);
   switch (a)
   {
      case 1 : b=2;
               break;
      case 2 : c=3;
                b=a*c;
               break;
      case 3 : {
                  c=a;
                }
      default: b=500;
   }
   printf("\nvalor de a = %d   valor de b = %d   valor de c = %d\n",
      a,b,c);
}
```

Vetores e Strings

1. Vetores

Vetores nada mais são que matrizes.

Matriz (Álgebra) -> Arranjo retangular de elementos de um conjunto.

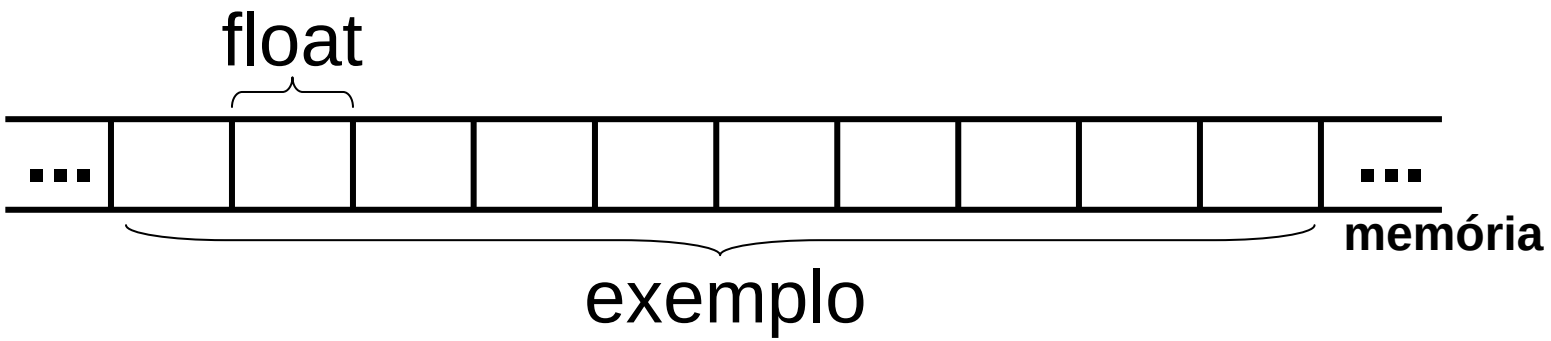
É importante notar que vetores, ou melhor, matrizes de qualquer dimensão são caracterizadas por terem todos os elementos pertencentes ao mesmo tipo de dado. Forma geral para se declarar um vetor unidimensional:

tipo_da_variavel nome_da_variavel [tamanho];

1. Vetores (continuação)

Exemplo:

float exemplo [10];



Vetores e Strings

1. Vetores (continuação)

Na linguagem C a numeração dos índices começa sempre em zero. Isto significa que, no exemplo acima, os dados serão indexados de 0 a 9. Para acessá-los vamos escrever:

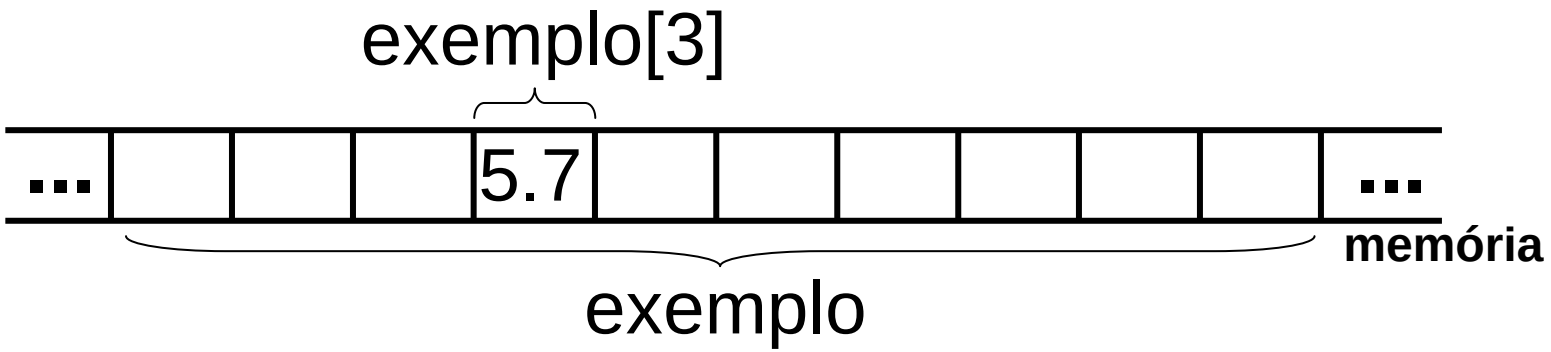
exemplo[**0**]
exemplo[**1**]
.
.
.
exemplo[**9**]

Vetores e Strings

1. Vetores (continuação)

Exemplo:

exemplo [3]=5.7;



Vetores e Strings

1. Vetores (continuação)

Mas ninguém o impede (programador) de escrever:

exemplo[30]

exemplo[103]

exemplo[-2]

Pois o C não verifica se o índice que você usou está dentro dos limites válidos. **Este é um cuidado que *você* deve tomar.** Se o programador não tiver atenção com os limites de validade para os índices ele corre o risco de ter variáveis sobreescritas ou de ver o computador travar. Inúmeros bugs podem surgir.

Vetores e Strings

1. Vetores (continuação)

Exercício:

Construa um programa que declare um vetor de inteiros com 10 elementos e o inicialize com números fornecidos pelo usuário, através da entrada padrão.

```
#include <stdio.h>
main()
{
   int vetor[10], indice;
   for (indice=0; indice<10; indice++)
   {
      printf("\nVetor[%d]: ",indice+1);
      scanf("%d",&vetor[indice]);
   }
}
```

Vetores e Strings

1. Vetores (continuação)

Exercício:

Construa um programa, com base no exercício anterior, que declare um vetor de inteiros com 10 elementos, o inicialize, com números fornecidos pelo usuário através da entrada padrão, e que através de uma pesquisa nos elementos do vetor, retorne na saída padrão os elementos de menor e maior valor, respectivamente.

```
#include <stdio.h>
main()
{
   int vetor[10], indice, menor, maior;
   for (indice=0; indice<10; indice++)
   {
      printf("\nVetor[%d]: ",indice+1);
      scanf("%d",&vetor[indice]);
   }
   for (indice=0;indice<10;indice++)
      if (!indice)
         menor=maior=vetor[indice];
      else
         if (menor>vetor[indice])
            menor=vetor[indice];
         else
            if (maior<vetor[indice])
               maior=vetor[indice];
   printf("\nO menor valor dos elementos do vetor eh: %d",menor);
   printf("\nO maior valor dos elementos do vetor eh: %d",maior);
}
```

Vetores e Strings

1. Vetores (continuação)

OBS.: Um vetor pode ser inicializado na declaração, exemplo:

```
int vetor[10]={0,1,2,3,4,5,6,7,8,9};
```

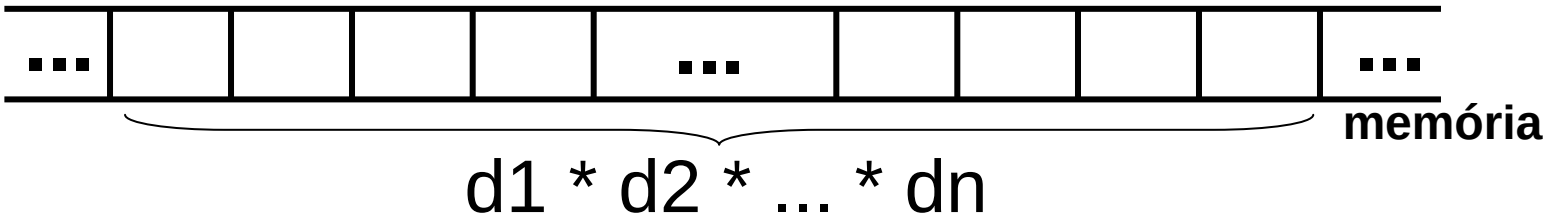
E ainda pode-se deixar em aberto o número de elementos, o qual será preenchido pelo número de valores fornecidos no momento da declaração, ou melhor, durante a inicialização.

```
int vetor[]={0,1,2,3,4,5,6,7,8,9};
```

~~int vetor[];~~ /* Não é permitido! */

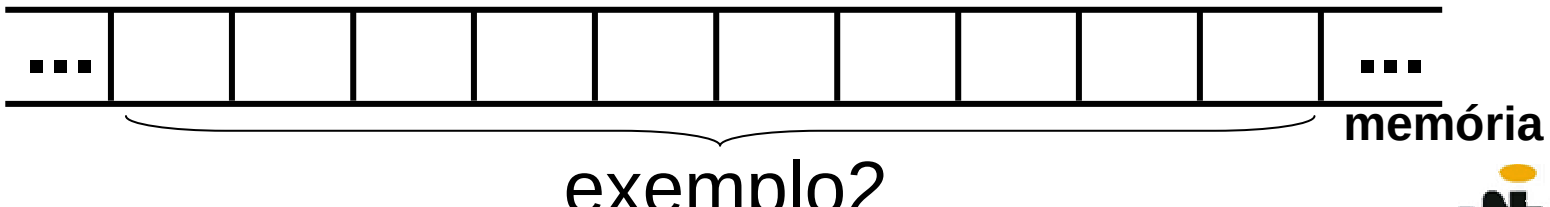
2. Vetores Multidimensionais

tipo_da_variavel nome_da_variavel [d1][d2]...[dn];



Exemplo:

int exemplo2 [2][5];

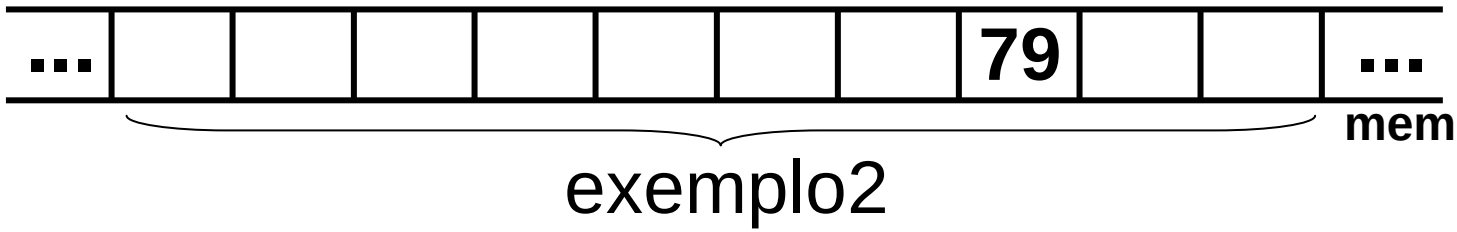


Vetores e Strings

Exemplo:

exemplo2 [1][2]=79;

O armazenamento de vetores multidimensionais se dá da seguinte forma: na primeira posição armazena-se o elemento com todos os índices zero, no caso do exemplo acima o elemento referenciado por exemplo2[0][0], seu sucessor é o elemento com o índice mais à direita incrementado em uma unidade (exemplo2[0][1]); quando o referido índice chegar ao seu valor máximo (exemplo2[0][4]) é incrementado o índice que o antecede e o seu valor é zerado (exemplo2[1][0]) e assim sucessivamente. Sendo assim, temos



Vetores e Strings

Exemplo:

O programa a baixo declara uma matriz 3x4 de inteiros e a inicializa com valores fornecidos pelo usuário.

```
#include <stdio.h>
#define n_l 3
#define n_c 4
main()
{
   int matriz[n_l][n_c], i, j;
   for (i=0;i<n_l;i++)
      for (j=0;j<n_c;j++)
      {
         printf ("\nEntre com matriz[%d][%d]=",i+1,j+1);
         scanf ("%d",&matriz[i][j]);
      }
}
```

Vetores e Strings

2. Vetores Multidimensionais (continuação)

Assim como os vetores unidimensionais os vetores multidimensionais também podem ser inicializados na declaração.

Exemplo:

float matriz [3][4]={1,2,3,4,5,6,7,8,9,10,11,12};

float matriz [ ][2]={1,2,3,4,5,6,7,8,9,10,11,12};

/* Uma dimensão pode ser omitida quando a inicialização se dá na declaração */

~~*float matriz [ ][ ]={1,2,3,4,5,6,7,8,9,10,11,12};*~~

/* Mais de uma dimensão não podem ser omitidas quando a inicialização se dá na declaração */

2. Vetores Multidimensionais (continuação)

Exercício: Construa um programa, na linguagem C, que declare uma matriz 7x4 de números em ponto flutuante, a inicialize com valores fornecidos pelo usuário através da entrada padrão e a retorne na saída padrão com o layout a seguir:

```
|     X.XX      X.XX ...          X.XX |
|-----10------|-----10-----| ...
|     X.XX      X.XX ...          X.XX |
|      .          .    .             .   |
|      .          .    .             .   |
|      .          .    .             .   |
```

Resposta trabalhada na Aula Prática X

```
#include <stdio.h>
#define nl 7
#define nc 4
main() {
   float matriz[nl][nc];
   int i, j;
   for (i=0;i<nl;i++)
      for (j=0;j<nc;j++) {
         printf ("\nEntre com matriz[%d][%d]=",i+1,j+1);
         scanf ("%f",&matriz[i][j]);
       }
   for (i=0;i<nl;i++) {
      printf("\n|");
      for (j=0;j<nc;j++)
         printf ("%10.2f",matriz[i][j]);
      printf(" |");
    }
}
```