

Estruturas de Controle de Fluxo

3. Laços de repetição

Exercício 12:

Construa um programa que leia da entrada padrão dois números naturais A e B, respectivamente, e retorne na saída padrão o quociente e o resto da divisão inteira de A por B. As únicas operações aritméticas que podem ser utilizadas na construção do programa são as de ***incremento*** e ***subtração***. Obs.: As entradas devem ser validadas.

A (dividendo)

B (divisor)

O quociente representa quantas vezes B cabe em A.
O resto representa a parte de A inferior a B.

Q=0; A>=B ? A=A-B; Q++;

A>=B ? A-=B; Q++;

...

A>=B ? Q (quociente) A (resto)

```
#include <stdio.h>
int main()
{
    int A, B, quociente=0;
    do {
        printf ("Entre com o valor natural para A (dividendo): ");
        scanf ("%d",&A);
    }while(A<0);
    do {
        printf ("Entre com o valor natural, exceto o zero para B (divisor): ");
        scanf ("%d",&B);
    }while(B<=0);
    while(A>=B) {
        A-=B;
        quociente++;
    }
    printf("\n0 quociente da divisao de A por B eh: %d",quociente);
    printf("\n0 resto da divisao de A por B eh: %d",A);
}
```

```
#include <stdio.h>
int main()
{
    int A, B, quociente;
    do {
        printf ("Entre com o valor natural para A (dividendo): ");
        scanf ("%d",&A);
    }while(A<0);
    do {
        printf ("Entre com o valor natural, exceto o zero para B (divisor): ");
        scanf ("%d",&B);
    }while(B<=0);
    for (quociente=0;A>=B; A-=B, quociente++);
    printf("\n0 quociente da divisao de A por B eh: %d", quociente);
    printf("\n0 resto da divisao de A por B eh: %d",A);
}
```

Estruturas de Controle de Fluxo

3. Laços de repetição

Exercício 13:

Construa um programa que leia da entrada padrão dois números inteiros A e B, respectivamente, e retorne na saída padrão o quociente e o resto da divisão inteira de A por B. O programa deve utilizar apenas as operações aritméticas de ***incremento***, ***decremento***, ***soma*** e ***subtração***.

A (dividendo) B (divisor)

+A/+B -A/-B +A/-B -A/+B

+15/+4

-15/-4

+15/-4

-15/+4

Q=0;

A>=B ?

A-=B;

Q++;

...

|A|>=|B| ?

A-=B;

Q++;

...

|A|>=|B| ?

A+=B;

Q--;

...

Q (quociente)

A (resto)

```
#include <stdio.h>
#include <math.h>
int main()
{
    int A,B,quociente;
    printf ("Entre com um valor inteiro para A (dividendo): ");
    scanf ("%d",&A);
    do {
        printf ("Entre com um valor inteiro para B (divisor): ");
        scanf ("%d",&B);
    }while (!B);
    if ((A>0 && B>0) || (A<0 && B<0))
        for (quociente=0; abs(A)>=abs(B); quociente++)
            A=A-B;
    else {
        for (quociente=0; abs(A)>=abs(B); quociente--)
            A=A+B;
    }
    printf("\n0 quociente da divisao eh: %d", quociente);
    printf("\n0 resto da divisao eh: %d",A);
}
```

Vetores

Vetores

1. Vetores

Vetores nada mais são que matrizes.

Matriz (Definição da 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 na linguagem C:

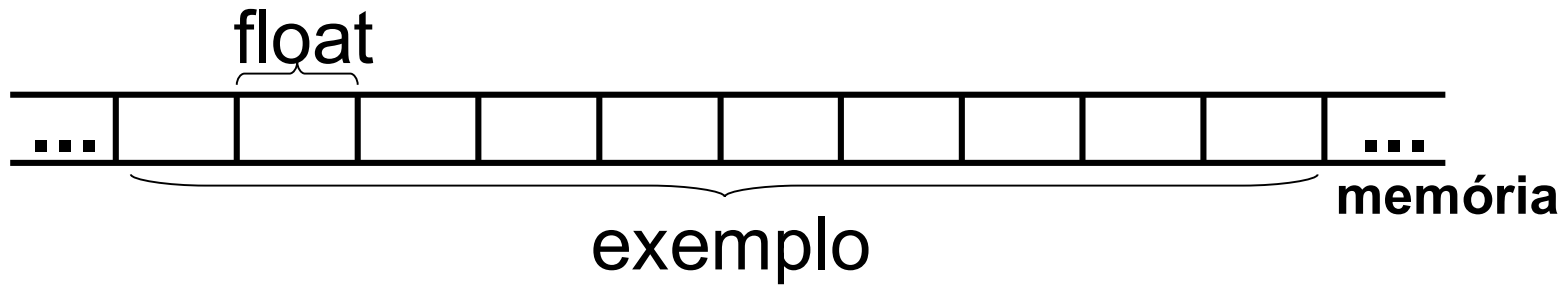
tipo_da_variavel nome_da_variavel [tamanho];

Vetores

1. Vetores

Exemplo:

float exemplo [10];



Vetores

1. Vetores

Se faz necessário acessar individualmente os elementos de um vetor.

Para isto utilizamos índices.

Na linguagem C a numeração dos índices começa sempre em zero. Isto significa que, no exemplo anterior (`float exemplo [10];`), os dados serão indexados de 0 a 9. Para acessá-los vamos escrever:

`exemplo[0]`

`exemplo[1]`

`...`

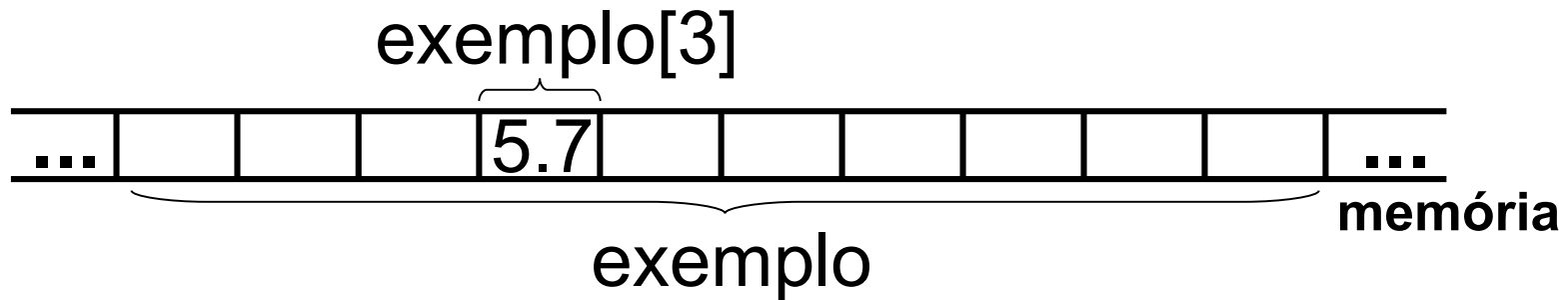
`exemplo[9]`

Vetores

1. Vetores

Exemplo:

exemplo [3]=5.7;



Vetores

1. Vetores

Mas, ninguém o impede (enquanto programador) de escrever:

```
exemplo[30]  
exemplo[103]  
exemplo[-2]
```

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

Vetores

1. Vetores

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[];~~

Vetores

1. Vetores

Exercício:

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

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

Vetores – Exercícios

Vetores

1. Vetores

Exercício:

Construa um programa, que declare um vetor de reais com 10 elementos, o inicialize, com números fornecidos através da entrada padrão, e, posteriormente, através de uma pesquisa nos elementos do vetor, retorne na saída padrão a posição no vetor do elemento com menor valor.

Observação: Considere que o vetor não possui valores iguais.

Exemplo de entrada:

2.6 0.0 9.2 7.3 98.0 99.9 -3.1 89.7 6.0 1.5

Saída para o exemplo de entrada:

7

```
#include <stdio.h>
int main() {
    float vetor[10], menor;
    int indice, ind_menor_ele;
    for (indice=0; indice<10; indice++)
    {   scanf("%f",&vetor[indice]);
for (indice=0; indice<10; indice++)
        if (!indice) {
            menor=vetor[indice];
            ind_menor_ele = indice;
        } else
            if (menor>vetor[indice]) {
                menor=vetor[indice];
                ind_menor_ele = indice;
            }
    }
}
```

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

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

Vetores

1. Vetores

Exercício:

Construa um programa, com base no exercício anterior, que declare um vetor de reais com 10 elementos, o inicialize, com números fornecidos através da entrada padrão, e, posteriormente através de uma pesquisa nos elementos do vetor, retorne na saída padrão a posição no vetor do elemento com menor valor.

Observação: Caso o vetor apresente valores iguais deve ser informada a maior posição dentre os valores iguais.

Exemplo de entrada:

2.6 0.0 9.2 -3.1 98.0 99.9 -3.1 9.2 6.0 1.5

Saída para o exemplo de entrada:

7

```
#include <stdio.h>
int main()
{
    float vetor[10];
    int indice, ind_menor_ele;
    for (indice=0; indice<10; indice++)
        scanf("%f",&vetor[indice]);
    for (ind_menor_ele=9, indice=8; indice>=0; indice--)
        if (vetor[ind_menor_ele]>vetor[indice])
            ind_menor_ele = indice;
    printf("%d", ind_menor_ele+1);
}
```