Aula 12

MC 102 - Algoritmos e Programação de Computadores

Vetores numéricos: definição e exemplos.

Como armazenar 3 notas?

```
float nota1, nota2, nota3;

printf("Nota do aluno 1: ");
scanf("%f", &nota1);

printf("Nota do aluno 2: ");
scanf("%f", &nota2);

printf("Nota do aluno 3: ");
scanf("%f", &nota3)
```

E se forem 100 notas?

```
float nota1, nota2, nota3, ..., nota100;

printf("Nota do aluno 1: ");
scanf("%f", &nota1);
printf("Nota do aluno 2: ");
scanf("%f", &nota2);
...
printf("Nota do aluno 100: ");
scanf("%f", &nota100)
```

E se forem n notas, com n<=100?

```
float nota1, nota2, nota3, ..., nota100;

if (n>=1){ printf("Nota do aluno 1: ");
          scanf("%f", &nota1); }
if (n>=2){ printf("Nota do aluno 2: ");
          scanf("%f", &nota2); }
...
if (n>=100){ printf("Nota do aluno 100: ");
          scanf("%f", &nota100);}
```

E se forem n notas, com n<=100?

```
float nota1, nota2, nota3, ..., nota100;

if (n>=1){ printf("Nota do aluno 1: ");
          scanf("%f", &nota1); }
if (n>=2){ printf("Nota do aluno 2: ");
          scanf("%f", &nota2); }
...
if (n>=100){ printf("Nota do aluno 100: ");
          scanf("%f", &nota100);}
```

Dá para fazer algo melhor?

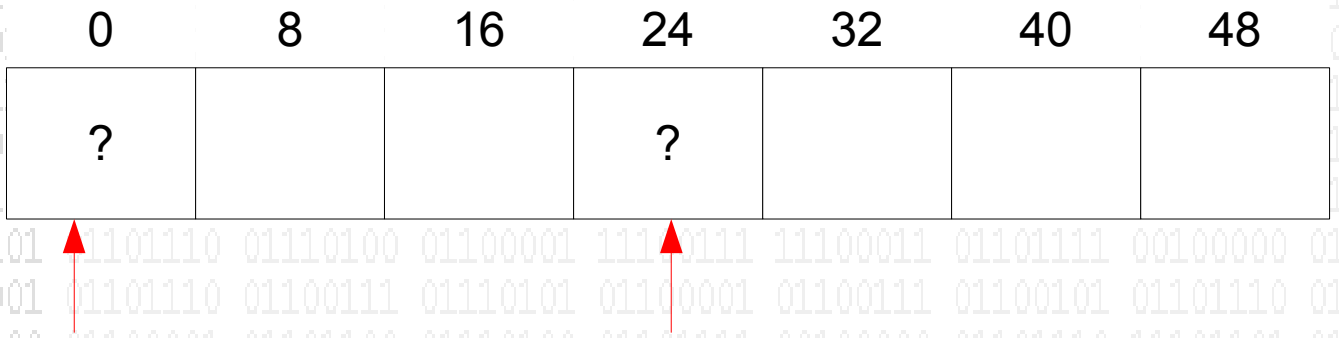
Vetor

é um tipo de dado usado para representar um conjunto (uma quantidade) definida de valores homogêneos.

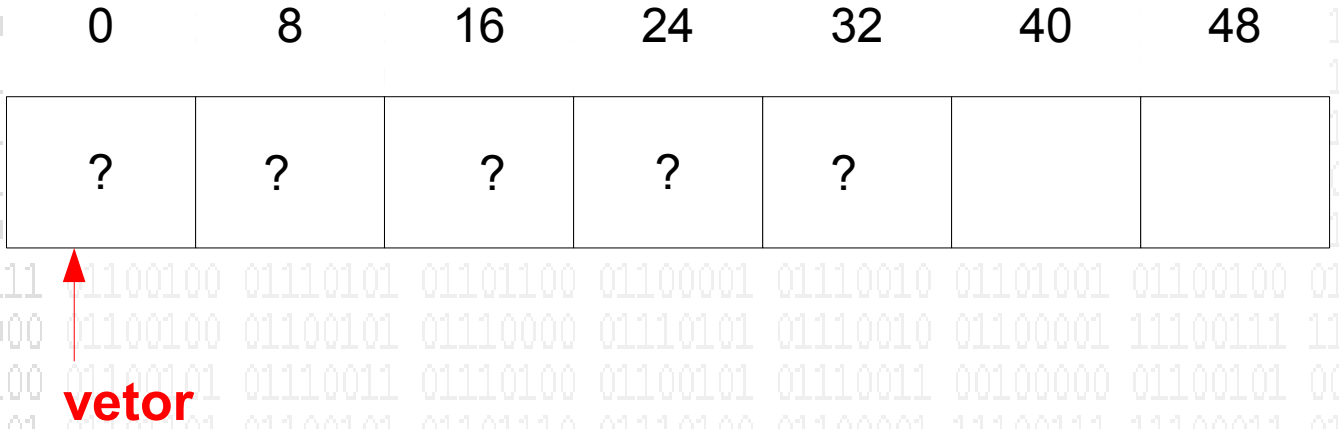
- Cada elemento é armazenado um ao lado do outro na memória
- Semelhante a um conjunto de variáveis do mesmo tipo, referenciadas por um mesmo nome de variável

Vetor

```
int a, b;
```




```
int vetor[5];
```



Como definir ?

Sintaxe:

tipo_de_dado identificador[numero_de_valores];

tipo_de_dado

char, int, short, float, etc...

identificador

Nome da variável para referência

numero_de_valores

Quantidade de valores que poderão ser armazenados

Exemplo:

```
float nota[5];
```

Atenção!!

- Cada posição do vetor é chamada de índice
- A primeira posição do vetor tem índice 0 (zero)
- Conseqüentemente, a última posição tem índice **N-1**, para **N** valores possíveis.
- Não se esqueçam disso para não terem problemas de "*falha de segmentação*".
- O tamanho do vetor, uma vez definido, não pode ser alterado

Como usar ?

- Definir o vetor

```
float nota[5];
```

- Atribuir valor

```
nota[2] = 7.6;
nota[1] = 8.1;
```

- Acessar valor armazenado

```
printf("Aluno %d - %.2f\n", 3, nota[2]);
```

- Atribuir valor de uma posição do vetor a uma variável

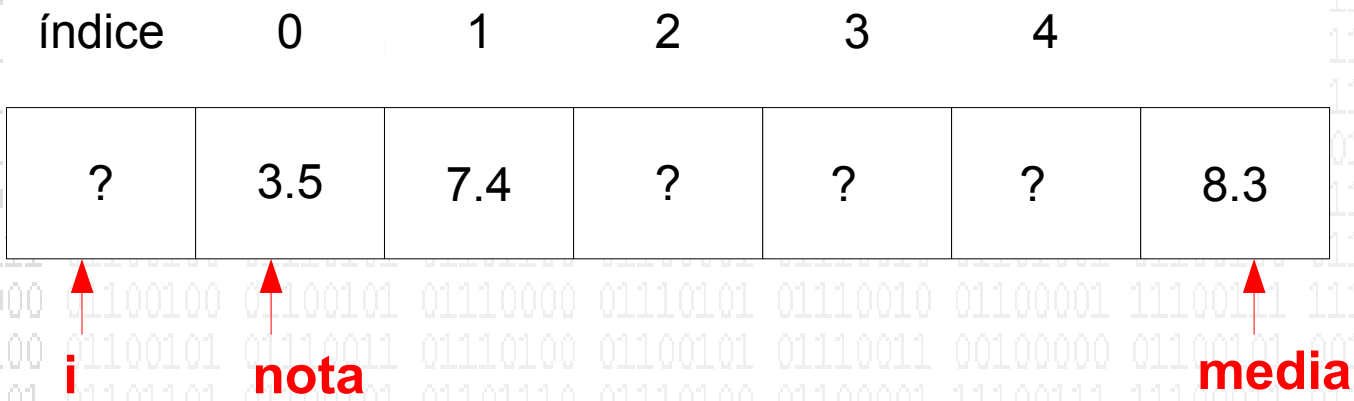
```
maior_nota = nota[1];
```

Usando vetor

```
#include <stdio.h>

int main(int argc, char **argv){
    int i;
    float not, nota[5], media;

    media = 0.0;
    for(i=0; i<5; i++) {
        printf("Nota do aluno %d:", i+1);
        scanf("%f", &not);
        nota[i] = not;
    }
    for(i=0; i<5; i++) {
        media += nota[i]/5;
    printf("média: %.2f\n", media);

    /* ... Outras contas usando o vetor nota ... */

 }
```

Vetor na Memória

```
float nota[5];

nota[0] = 3.5;
nota[1] = 7.4;
nota[5] = 8.3;
```

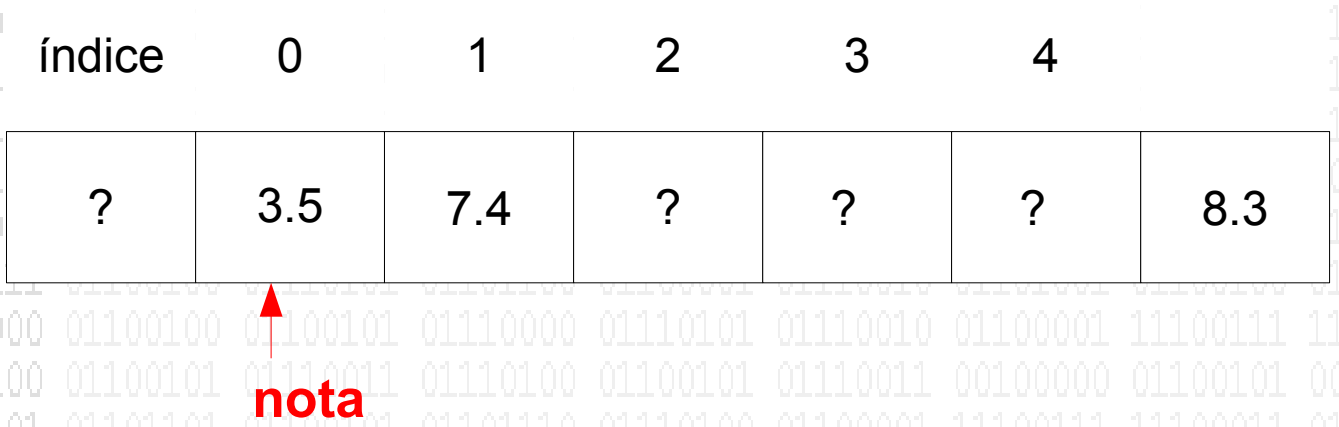


Na prática...

Em memória, um vetor de N posições ocupa o espaço equivalente a N variáveis do tipo de dado especificado.

Cada espaço reservado para um valor é equivalente a uma variável, mas está associado a um índice do vetor, de 0 até N-1, portanto também tem um endereço de memória como qualquer outra variável.

O que significa `&nota[0]` ? Pode-se fazer `&nota[3]` ?



Endereços de memória

Para fazer a leitura do teclado usando scanf é necessário utilizar o endereço de memória da variável.

Para vetores, é a mesma coisa:

- `nota` é equivalente a `&nota[0]`

  Endereço da primeira posição do vetor

- `&nota[3]` é o endereço da 4a. posição

  Em loops, você pode usar `&nota[i]` para a i-ésima posição

Principais Vantagens

Usar um único identificador para representar um conjunto de variáveis e facilitar o seu manuseio.

É muito útil quando precisamos armazenar um número indeterminado (mas finito) de valores.

Inicialização de vetores

Da mesma forma que variáveis, pode-se definir os valores iniciais para cada posição de um vetor. Porém, isso só é possível na **definição do vetor**. Caso contrário, é preciso inicializar cada posição individualmente.

Inicializando o vetor nota:

```
float nota[5] = {0.0, 0.0, 0.0, 0.0, 0.0};
```

Inicialização de cada posição do vetor:

```
for(i=0;i<5;i++)
   nota[i] = 0.0;
```

Programa 1

Escreva um programa que leia 10 números e retorne o valor máximo, o mínimo e imprima a diferença de cada valor pelo valor mínimo encontrado. Use um loop separado para encontrar o máximo e o mínimo.

Programa 1 - Algoritmo

Entrada de dados:

10 números

Algoritmo:

Ler os 10 números

Encontrar o máximo (max) e o mínimo (min)

Imprimir max, min e (valor – min) para cada valor

Programa 1 – Passo 1

Entrada de dados:

10 números

Algoritmo:

Ler os 10 números

Para i de 1 até 10

Ler o i-ésimo número

Encontrar o máximo e o mínimo

Imprimir max, min e (valor – min) para cada valor

Programa 1 – Passo 2

Entrada de dados:

10 números

Algoritmo:

Ler os 10 números

Encontrar o máximo e o mínimo

```
max = min = 1o. número
Para i de 2 até 10
    Se i-ésimo número > max
        max = i-ésimo número
    Senão, se i-ésimo número < min
        min = i-ésimo número
```

Imprimir max, min e (valor – min) para cada valor

Programa 1 – Passo 3

Entrada de dados:

10 números

Algoritmo:

Ler os 10 números

Encontrar o máximo e o mínimo

Imprimir max, min e (valor – min) para cada valor

```
imprime max e min
Para i de 1 a 10
    imprime i-ésimo número - min
```

Programa 1 - Código

```
#include <stdio.h>

int main(int argc, char **argv) {
  float numero[10], max, min;
  int i;

  printf("Informe os 10 números:\n");
  for (i=0; i<10; i++)
    scanf("%f", &numero[i]);

  max = min = numero[0];
  for (i=1; i<10; i++)
    if (numero[i] > max)
      max = numero[i];
    else if (numero[i] < min)
      min = numero[i];

  printf("Maximo: %f\tMinimo: %f\n", max, min);
  for(i=0; i<10; i++)
    printf("%f\n", numero[i] - min);
}
```

Programa 2

Faça um programa que, dados 10 números inteiros positivos, retorne os valores normalizados no intervalo [0,100].

Para realizar a normalização, é necessário saber o maior valor dos números informados. Calcule esse máximo em um loop separado.

Programa 2 - Algoritmo

Entrada de dados:

10 números

Algoritmo:

Ler os 10 números

Encontrar o máximo

Para cada número, calcular 100*numero/máximo

Imprimir valores calculados

Programa 2 - Código

```
#include <stdio.h>
#define MAIOR_VALOR 100

int main(int argc, char **argv) {
  unsigned int i, numero[10], max;

  printf("Informe os 10 números:\n");
  for (i=0; i<10; i++)
    scanf("%u", &numero[i]);

  max = numero[0];
  for (i=1; i<10; i++)
    if (numero[i] > max)
      max = numero[i];

  for (i=0; i<10; i++)
    numero[i] = MAIOR_VALOR * ((float) numero[i] / max);

  for(i=0; i<10; i++)
    printf("%u\n", numero[i]);
}
```

Programa 3 - Histograma

Escreva um programa que leia 65 notas do teclado e retorne a freqüência de ocorrência das notas nos intervalos inteiros:

[0,1), [1,2), [2,3), [3,4), [4,5), [5,6), [6,7), [7,8), [8,9) e [9,10]

Utilize apenas um vetor para armazenar a quantidade de notas nesses intervalos.

Programa 3 - Algoritmo

Entrada de dados:

65 notas

Algoritmo:

Para i de 1 a 65

Informe i-ésima nota
posição = parte inteira da nota
Se intervalo >= 10
intervalo = 9
Acumula 1 em histograma para intervalo

Imprimir valores do histograma

Programa 3 - Código

```
#include <stdio.h>
#define NUMERO_NOTAS 65

int main(int argc, char **argv) {
  float nota;
  int i, intervalo, hist[10] = {0,0,0,0,0,0,0,0,0,0};

  printf("Informe as 65 notas:\n");
  for (i=0; i < NUMERO_NOTAS; i++) {
    scanf("%f", &nota);
    intervalo = nota;
    if (intervalo >= 10)
      intervalo = 9;
    hist[intervalo]++;
  }

  printf("Histograma:\n");
  for(i=0; i < 10; i++)
    if (i < 9)
      printf("[%2d,%2d) = %d\n", i, i+1, hist[i]);
    else
  printf("[%2d,%2d] = %d\n", i, i+1, hist[i]);
}
```

Tarefa

Refaça o programa do troco utilizando agora **dois vetores**. Um para moedas, inicializado com o valor de cada moeda, e um para a quantidade necessária de cada moeda.

Tarefa: Escreva um programa (usando vetores) que encontre o menor número possível de moedas para formar uma quantia Q de dinheiro, um **inteiro positivo** informado em centavos. Existem apenas as moedas de 1, 5, 10, 25, 50 e 100 centavos. Imprimir o número de moedas necessárias de cada tipo.