Aula 13
Vetores

Roteiro

Vetores

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);
```

Vetores

Como armazenar 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);
```

Vetores — Definição

- Coleção de variáveis do mesmo tipo referenciada por um nome comum (Herbert Schildt).
- Características:
  - Acesso por meio de um índice inteiro.
  - Posições contíguas na memória.
  - Tamanho pré-definido.
  - Índices fora dos limites podem causar comportamento anômalo do programa.

Declaração de um vetor

<tipo> identificador [<tamanho do vetor>];

Exemplo

```
float notas[100];
int primos[100];
```

Usando um vetor

Após declarada uma variável do tipo vetor, pode-se acessar uma determinada posição do vetor utilizando um valor inteiro.

identificador [<posição>];

- A primeira posição de um vetor tem índice 0.
- A última posição de um vetor tem índice *<tamanho do vetor>* - 1.

O vetor em uma posição específica tem o mesmo comportamento que uma variável simples.

Exemplo

```
int nota[10];
int a;
nota[5] = 95;
a = nota[5];
```

Usando um vetor

identificador [<posição>];

- Você pode usar valores inteiros para acessar uma posição do vetor.
- O valor pode ser inclusive uma variável inteira.

Exemplo

```
int g, vet[10];
for(g=0; g<10; g++)
    vet[g]=5*g;
```

Vetores

- Na memória:

```
int d;
int vetor[5];
int f;
```

Nome	d	vetor					f
Índice	-	0	1	2	3	4	-

Vetores

- Ao executar `vetor[3]=10;`:

Nome	d	vetor					f
Índice	-	0	1	2	3	4	-
					10		

Vetores

- O que ocorre se for executado os comandos:
  `vetor[5]=5;`
  `vetor[-1]=1;`

Vetores

- Ao executar

```
vetor[3]=10;
vetor[5]=5;
vetor[-1]=1;
```

Nome	d	vetor					f
Índice	-	0	1	2	3	4	-
	1				10		5

- Isto irá causar um erro no seu programa pois você está alterando valores de outras variáveis.
- Em muitos casos o seu programa será encerrado (**Segmentation Fault**).

Questões importantes sobre vetores

- O tamanho do vetor é pré-definido (durante a execução do programa não pode ser alterado).
- Índices fora dos limites podem causar comportamento anômalo do código.

Como armazenar n (<= 100) notas?

```
float nota[100];
int n, i;

printf("Número de alunos: ");
scanf("%d", &n);

for (i = 0; i < n; i++) {
  printf("Nota do aluno %d: ", i);
  scanf("%f", &nota[i]);
}
```

- O programa acima está correto?

Como armazenar n (<= 100) notas?

- Você deve testar se $n > 100$ para evitar erros!!

```
float nota[100];
int n, i;

printf("Número de alunos: ");
scanf("%d", &n);
if(n>100){
  n=100;
  printf("\nNumero maximo de alunos alterado para 100");
}
for (i = 0; i < n; i++) {
  printf("Nota do aluno %d: ", i);
  scanf("%f", &nota[i]);
}
```

Exemplo: Produto Interno de dois vetores

- Ler dois vetores de dimensão 5 e computar o produto interno destes.
- Quais tipos de variáveis usar?

Exemplo: Produto Interno de dois vetores

Ler dois vetores de dimensão 5 e computar o produto interno destes.

```
int main(){
  double vetor1[5], vetor2[5], resultado;
  int i;

  for(i=0; i<5; i++){
    printf("Entre com valor %d para vetor 1:",i+1);
    scanf("%lf",&vetor1[i]);
  }
  printf("\n\n");
  for(i=0; i<5; i++){
    printf("Entre com valor %d para vetor 2:",i+1);
    scanf("%lf",&vetor2[i]);
  }

  //calculando o produto interno
  resultado = 0.0;
  for(i=0; i < 5; i++){
    resultado = resultado + ( vetor1[i]*vetor2[i] );
  }
  printf("\n\nO produto interno e: %lf\n",resultado);
}
```

Exemplo: Elementos Iguais

- Ler dois vetores com 5 inteiros cada.
- Checar quais elementos do segundo vetor são iguais a algum elemento do primeiro vetor.
- Se não houver elementos em comum, o programa deve informar isso.

Exemplo: Elementos Iguais

```
int main(){
  int vetor1[5], vetor2[5];
  int i, j,  umEmComum;

  for(i=0; i<5; i++){
    printf("Entre com valor %d para vetor 1:",i+1);
    scanf("%d",&vetor1[i]);
  }
  printf("\n\n");
  for(i=0; i<5; i++){
    printf("Entre com valor %d para vetor 2:",i+1);
    scanf("%d",&vetor2[i]);
  }

  umEmComum = 0;
  for(i = 0; i < 5 ; i++)
    for(j = 0; j < 5; j++)
      if(vetor1[i] == vetor2[j]){
         umEmComum = 1;
         printf("Posicao %d do vetor1 igual a %d do vetor2.\n",i,j);
      }
  if(!umEmComum)
    printf("Nenhum elemento em comum!\n");
}
```

Vetores em funções

- Vetores também podem ser passados como parâmetros em funções.
- Ao contrário dos tipos simples, vetores têm um comportamento diferente quando usados como parâmetros de funções.
- Quando uma variável simples é passada como parâmetro, seu valor é atribuído para uma nova variável local da função.
- No caso de vetores, **não é criado** um novo vetor!
- Isto significa que os valores de um vetor **são alterados** dentro de uma função!

Vetores em funções

```
#include <stdio.h>

void fun1(int vet[], int tam){
  int i;
  for(i=0;i<tam;i++)
     vet[i]=5;
}

int main(){
  int x[10];
  int i;

  for(i=0;i<10;i++)
    x[i]=8;

  fun1(x,10);
  for(i=0;i<10;i++)
    printf("%d\n",x[i]);
}
```

Vetores em funções

- Vetores não podem ser devolvidos por funções.
- Mas mesmo assim, podemos obter um resultado parecido com isso, usando o fato de que vetores são alterados dentro de funções.

```
#include <stdio.h>

int[] leVet() {
  int i, vet[100];
  for (i = 0; i < 100; i++) {
    printf("Digite um numero:");
    scanf("%d", &vet[i]);
  }
}
```

O código acima não compila, pois não podemos retornar um **int[]** .

Vetores em funções

- Mas como um vetor é alterado dentro de uma função, podemos criar a seguinte função:

```
#include <stdio.h>

void leVet(int vet[], int tam){
  int i;
  for(i = 0; i < tam; i++){
    printf("Digite numero:");
    scanf("%d",&vet[i]);
  }
}

void escreveVet(int vet[], int tam){
  int i;
  for(i=0; i< tam; i++)
    printf("vet[%d] = %d\n",i,vet[i]);
}
```

Vetores em funções

```
int main(){
  int vet1[10], vet2[20];

  printf(" ------ Vetor 1 --------\n");
  leVet(vet1,10);
  printf(" ------ Vetor 2 --------\n");
  leVet(vet2,20);

  printf(" ------ Vetor 1 --------\n");
  escreveVet(vet1,10);
  printf(" ------ Vetor 2 --------\n");
  escreveVet(vet2,20);

}
```

Exercício

- Crie uma função
  int maiorValor(int vet[], int tam)
  que recebe como parâmetros um vetor e seu tamanho e devolve o maior valor armazenado no vetor.

Exercício

- Crie uma função
  double media(int vet[], int tam)
  que recebe como parâmetros um vetor e seu tamanho e devolve a média dos valores armazenados no vetor.

Exercício

- Crie uma função
  int checa(int vet[], int tam, int C)
  que recebe como parâmetros um vetor, seu tamanho e um inteiro C. A função deve retornar 1 caso existam dois elementos distintos do vetor tal que a multiplicação destes é C.
- Exemplo: Se $vet = (2, 4, 5, -10, 7)$ e $C = 35$ então a função deve devolver 1. Mas se $C = -1$ então a função deve devolver 0.

Informações Extras: Inicialização de um vetor

- Em algumas situações é necessário declarar e já atribuir um conjunto de valores contantes para um vetor.
- Em C, isto é feito atribuindo-se uma lista de elementos para o vetor na sua criação da seguinte forma:

<tipo> identificador [] = {elementos separados por vírgula} ;

- Exemplos:

```
double vet1[] = {2.3, 3.4, 4.5, 5.6};
int vet2[] = {5, 4, 3, 10, -1, 0};
```

Informações Extras: Inicialização de um vetor

```
#include <stdio.h>

int main(){
  double vet1[] = {2.3, 3.4, 4.5, 5.6};
  int vet2[] = {5, 4, 3, 10, -1, 0};
  int i;

  for(i=0; i<4; i++)
    printf("%lf\n", vet1[i]);

  for(i=0; i<6; i++)
    printf("%d\n", vet2[i]);

}
```