Vetores

Professores(as):
Virgínia Fernandes Mota
João Eduardo Montandon de Araujo Filho
Leandro Maia Silva

Estruturas de Dados

- Em diversas situações os tipos básicos de dados (inteiro, real, caracter, ...) não são suficientes para representar a informação que se deseja armazenar.
  - Exemplo, uma palavra: "AULA".
  - Valor de 50 produtos em uma loja
- Existe a possibilidade de construção de novos tipos de dados a partir da composição (ou abstração) de tipos de dados primitivos.
- Esses novos tipos tem um formato denominado ESTRUTURA DE DADOS, que define como os tipos primitivos estão organizados.

Motivação

- Problema 1: Como poderíamos fazer um algoritmo para ler 50 notas de uma turma e calcular sua média?

```c
#include <stdio.h>
#define SUCESSO 0

int main(int argc, char ** argv){
   int i;
   float media, nota, soma = 0.0;

   for(i = 0; i < 50; i++){
      printf("Digite sua nota: ");
      scanf("%f",&nota);
      soma += nota;
   }

   media = soma / 50;
   printf("%f",media);

   return SUCESSO;
}
```

- Mas e se eu precisar saber o valor da 1a, 4a ou 40a nota?

Motivação

- Problema 2: Fazer um programa para ler 50 notas de uma turma e calcular a sua média. Imprimir as notas lidas juntamente com a média da turma como na tabela.

Nota	Média
8.0	7.75
4.6	7.75
2.3	7.75
7.8	7.75
...	...

- Como fazê-lo? No exemplo anterior, uma nota é sobreposta por outra em cada iteração do for.
- A solução é armazenar todas as 50 notas lidas... Como?

Variáveis Compostas

- Conjunto de variáveis identificadas por um mesmo **nome**.
  - **Homogêneas:** vetores e matrizes;
  - **Heterogênea:** estruturas.

Variáveis Compostas Homogêneas

- Quando uma determinada estrutura de dados for **composta de variáveis com o mesmo tipo**, temos um conjunto homogêneo de dados;
- As **variáveis compostas homogêneas unidimensionais** são utilizadas para representar arranjos unidimensionais de elementos de um mesmo tipo, em outras palavras, são utilizadas para representar **vetores**;

Vetores

- Declaração:
  tipo identificador[n_elementos]
  - tipo: um dos tipos de dados em C (char, int, float, double, ...);
  - identificador: segue as mesmas regras das variáveis básicas;
  - n_elementos: define o número máximo de elementos do vetor. Também pode ser uma expressão constante inteira.

- Exemplo: vetor com 5 elementos do tipo inteiro
  int dados[5]; //5 elementos quaisquer do tipo inteiro.

Posição:	0	1	2	3	4
Valor:	?	?	?	?	?

- Acesso: vetor[posicao]
- Exemplo: dados[3]

Vetores Numéricos

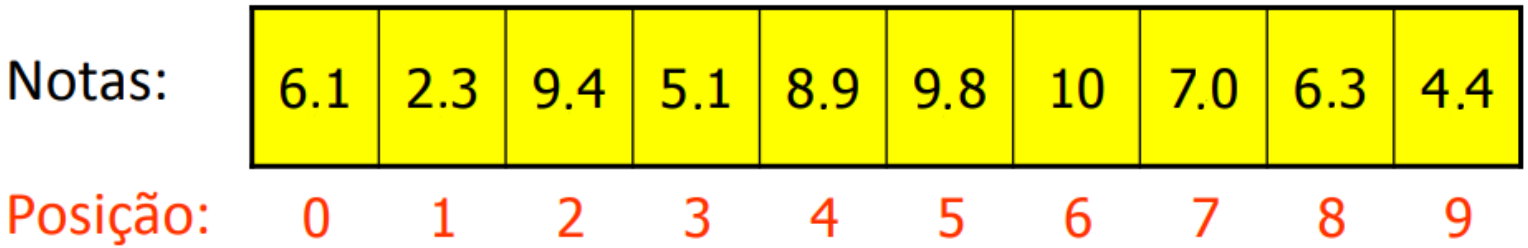
- Recebem valores inteiros, de ponto flutuante (Precisão Simples) - float e de ponto flutuante (Precisão Dupla) - double;
- Declaração e Inicialização (feitas conjuntamente):

```
1 /* inicializa todos com 0*/
2 int Vet1[4] = {0,0,0,0};
3 /* inicializa os dois primeiros elementos com -1*/
4 int Vet2[4] = {-1,-1};
5 /* inicializa todos com valores tipo float*/
6 float Vet3[3] = {1.0f, 1.1f, 1.5f};
7 /* a dimensao assume o tamanho da inicialização*/
8 int Vet4[ ] = {0,0,0,0,0,0,0,0};
```

- A declaração e inicialização conjuntas é útil para vetores de dimensão reduzida.

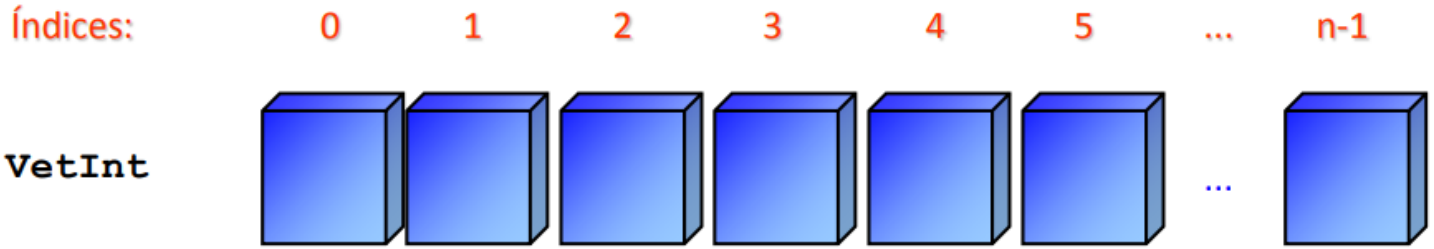
Vetores Numéricos

- Correspondem a posições da memória:
  - identificadas por um único nome;
  - individualizadas por índices;
  - cujo conteúdo é de um mesmo tipo.



Vetores Numéricos

```
int VetInt[n];
```



Índice do primeiro elemento: zero
Índice do último elemento: $n - 1$
Quantidade de elementos: n

Vetores Numéricos - Exemplos

- O programa a seguir, usa o comando for para inicializar com zeros os elementos de um array (vetor) inteiro n de 10 elementos e o imprime sob a forma de uma tabela.

```
1  #include <stdio.h>
2  #define SUCESSO 0
3  #define TAMANHO_VETOR  10
4
5  int main(int argc, char ** argv){
6      int n[TAMANHO_VETOR], i;
7
8      //inicializacao do vetor
9      for(i = 0; i< TAMANHO_VETOR; i++){
10         n[i] = 0;
11     }
12
13     printf("Elemento \t Valor \n");
14     for(i = 0; i< TAMANHO_VETOR; i++){
15         printf("%d \t %d \n", i, n[i]);
16     }
17
18     return SUCESSO;
19 }
```

Vetores Numéricos - Exemplos

- O programa abaixo inicializa os dez elementos de um array s com os valores: 2, 4, 6, ..., 20 e imprime o array em um formato de tabela.

```
1  #include <stdio.h>
2  #define SUCESSO 0
3  #define TAMANHO_VETOR 10
4
5  int main(int argc, char ** argv){
6      int s[TAMANHO_VETOR], i;
7
8      //inicializacao do vetor
9      for(i = 0; i < TAMANHO_VETOR; i++){
10         s[i] = 2 + 2*i;
11     }
12
13     printf("Elemento \t Valor \n");
14     for(i = 0; i< TAMANHO_VETOR; i++){
15         printf("%d \t %d \n", i, s[i]);
16     }
17
18     return SUCESSO;
19 }
```

Alocação estática X Alocação dinâmica

- Até agora vimos como alocar um espaço estático para as variáveis.
- A alocação estática é uma estratégia de alocação de memória na qual toda a memória que um tipo de dados pode vir a necessitar é alocada de uma vez.
  - int v[10]; $\rightarrow$ aloca um espaço contíguo de 10 valores inteiros
- Mas e se eu não souber o tamanho que devo alocar?

Alocação estática X Alocação dinâmica

- A alocação dinâmica é uma técnica que aloca a memória sob demanda.
- Os endereços podem ser alocados, liberados e realocados para diferentes propósitos, durante a execução do programa.
- Em C usamos **malloc(n)**, ou similar, para alocar um bloco de memória de tamanho n bytes.
- É responsabilidade do programador de liberar a memória após seu uso.
- Veremos como utilizar a alocação dinâmica mais adiante no curso!!

Exercícios

1. Faça um programa que leia, via teclado, 20 valores do tipo inteiro e determine qual o menor valor existente no vetor e imprima o valor e seu índice no vetor;
2. Desenvolva um programa que leia um vetor de números reais, um escalar e imprima o resultado da multiplicação do vetor pelo escalar;
3. Faça um programa que leia 2 vetores de tamanho 10 e calcule o produto escalar deles;
4. Faça um programa que leia um vetor de um tamanho escolhido pelo usuário e calcule a média aritmética de seus valores.