Estrutura de Dados Homogêneos e Heterogêneos

Profª. Rose Yuri Shimizu

Roteiro

Estrutura de Dados Homogêneos: Array

- Estrutura de Dados Homogêneos
  - Organizam uma coleção de dados
  - Dados do mesmo tipo
- Elementar (lista utilizada por outras estruturas)
- Tamanho fixo ou variado (depende da alocação)
- Dimensões
  - Unidimensional: vetor
  - Multdimensional: matriz

Vetor ou Array unidimensional - declaração

- 1 variável → 1 tipo → vários conteúdos
- Declaração:

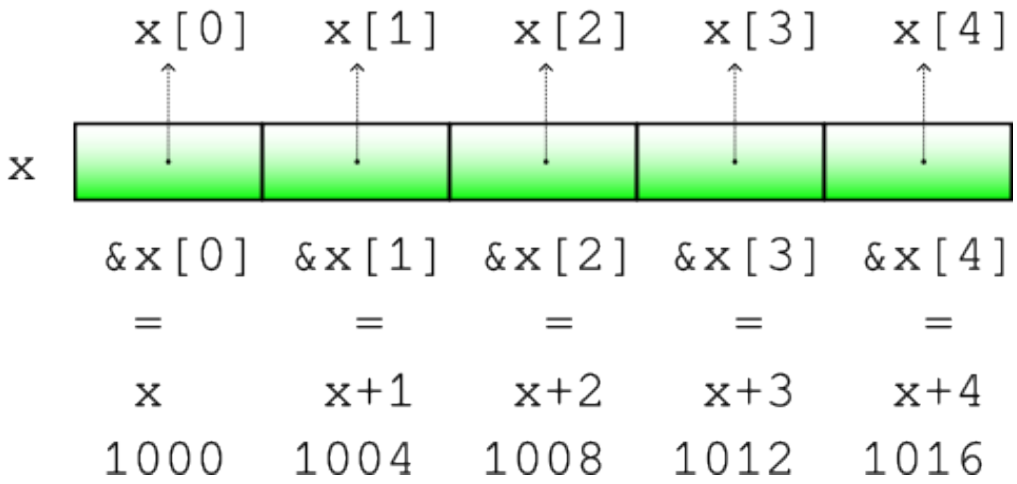
```
1 //tipo variavel[tamanho];
2 int produtos[5];
3 float precos[3];
4 char palavra[50];
5
6 //variaveis como indices
7 int i=5;
8 int x[i];
```

```
int a[5];
```

Vetor ou Array unidimensional - endereço

- Variável = endereço na memória
- Acesso aos elementos
  - CADA POSIÇÃO: 0 $\rightarrow$ n-1
  - Aleatório: qualquer posição pode ser acessada facilmente através de um **index**
  - Cada elemento: endereço sequencial a partir da primeira posição
  - `x = &x[0]`

```
int x[5];
```



Vetor ou Array unidimensional - inicialização na declaração

```
1  //Inicialização na declaração
2  float dinheiro[3] = {23.4, 123.0, 55.0};
3  char letras[4] = {'a', 'b', 'c', 'd'};
4
5  int a=5;
6  int y[a] = { 1, 4, 6, 99, 2}; //erro de compilação
7
8  int erro[5];
9  erro = { 2, 4, 6, 8, 10 }; //erro
10
11 int peso[] = { 153, 135, 170 };
12 int b[]; //erro
```

```
1  int a[6]={10,20,40,45,30,12};
```

Vetor ou Array unidimensional - inicialização pela entrada

```c
float dinheiro[100]; //0 -> 99
char letras[4]; //0 -> 3
int i=0;

while(i<100) {
    scanf("%f", &dinheiro[i]);
    i++;
}

for(i=0; i<4; i++) {
    scanf(" %c", &letras[i]);
        // ^ espaço em leituras seguidas de caracteres
}

dinheiro[2] = 1;  //acesso direto e aleatório
int x = dinheiro[3];

dinheiro[2]++;        //operações
```

Vetor ou Array unidimensional - inicialização por cópia

```
1  int vetorA[10], vetorB[10];
2  for (int i = 0; i < 10; i++) {
3      scanf("%d", &vetorA[i]);
4  }
5
6  // copiar o conteudo do vetor A para o vetor B
7  vetorB = vetorA;    //????
8
9  for (int i = 0; i < 10; i++) {
10     vetorB[i] = vetorA[i];
11 }
```

Vetor ou Array unidimensional - operações

Busca

```
1 //funciona
2 int achou = 0, k = 0;
3 while (k < n && achou == 0) { //testa condicao
4     if (v[k] == x) achou = 1;  //testa condicao (mais uma)
5     else k++;
6 }
```

Vetor ou Array unidimensional - operações

Busca

```
1 int k = 0;
2 // unifica as condicoes
3 while (k < n && v[k] != x) k++;
```

```
1 int k = 0;
2 // problema?!
3 while (v[k] != x && k < n) k++;
```

```
1 int v[6] = {1, 3, 2, 5, 6}; //uma posicao a mais
2 int k = 0, x = 2;
3
4 v[5] = x;  // sentinela
5 while (v[k] != x) k++;
6
7 //encontrou?!?!
```

Vetor ou Array unidimensional - operações

```c
/*  Procura x em um vetor v de tamanho n */
int busca_v(int x, int v[], int n)
{
    int k = n-1;
    while(k >= 0 && v[k] != x) k--;

    //funciona?
    //custo: quantos elementos precisam ser percorridos?

    return k;
}
```

Vetor ou Array unidimensional - operações

```c
//Remove o elemento na posição k do vetor v de tamanho n
int remove_v(int k, int v[], int n)
{
    int x = v[k], j;

    //puxando
    for(j = k+1; j < n; ++j)
        v[j-1] = v[j];

    //funciona?
    //custo: quantos elementos precisam ser percorridos?

    return x;
}
```

Vetor ou Array unidimensional - operações

```c
/*  Insere x na posição k do vetor v de tamanho n
    Vetor não pode estar cheio  */
void insere_v(int x, int k, int v[], int n)
{
    //puxando - fim para o início (???)
    for (int j = n-1; j > k; --j)
        v[j] = v[j-1];

    v[k] = x;

    //e sem deslocamento, quantos movimentos?
}
```

String - Sequência de caracteres - declaração e inicialização

- São arrays (vetores) de caracteres (char) terminados com **'\0'**.
- O compilador automaticamente coloca o '\0'
- Lembre-se: sempre adicionar 1 no tamanho do array

```
1  //Inicialização na declaração
2
3  //string - fim bem definido
4  char nome1[6] = "maria";
5  char nome2[6] = {'m', 'a', 'r', 'i', 'a', '\0'};
6
7  //vetor de caracteres
8  char nome3[5] = {'m', 'a', 'r', 'i', 'a'};
9
10 //especificador de formato: %s
11 printf("%s %s %s\n", nome1, nome2, nome3); //maria maria maria
```

String - Sequência de caracteres - inicialização pela entrada

```c
char nome1[100];
char nome2[100];
char c;

//Lendo uma palavra em uma string
scanf("%s", nome1); //sem & : vetor = endereco da 1a posicao
                    //lê 1(uma) seq. de caracteres (palavra)
                    //ate encontar ' ', \t, \n
                    //scanf coloca \0 no final

//Lendo uma palavra em um vetor de caracteres
scanf(" %c", &c); //por que o ' ' antes do %c?
for(int i=0; i<100 && c!=' ' && c!='\n'  && c!='\t'; i++){
    nome2[i] = c;
    scanf("%c", &c); //por que não tem ' ' antes do %c?
}

//especificador de formato: %s
printf("%s e %s\n", nome1, nome2);
```

String - Sequência de caracteres - leitura formatada

```c
char nome1[100];
char nome2[100];
char nome3[100];

scanf("%s", nome1); //sem & : vetor = endereco da 1a posicao
                    //lendo 1 sequencia de caracteres (palavra)
                    //ate encontar ' ', \t, \n

scanf("%99s", nome2); //le no maximo 99 caracteres seguidos
                      //ou ate encontrar ' ', \t, \n

scanf("Estrutura de %s", nome3);

//especificador de formato: %s
printf("%s %s %s\n", nome1, nome2, nome3);
```

- Dadas as entradas:
  - João
  - José Paulo
  - Estrutura de Dados

- Qual a saída? 1, 2, 3, 4, 5 ou 6?
  1. João José
  2. João José Dados
  3. João José Estrutura de Dados
  4. João José Paulo
  5. João José Paulo Dados
  6. João José Paulo Estrutura de Dados

String - Sequência de caracteres - frases

```c
char nome1[100];

//leia tudo menos a quebra de linha
//leia tudo até a quebra de linha
scanf("%99[^\n]", nome1);

printf("%s\n", nome1);
```

Exemplo:

Entradas	**Saída:**
Jose da Silva	Jose da Silva

String - Sequência de caracteres - leituras seguidas

```c
char nome1[100], nome2[100], nome3[100], nome4[100];

scanf("%99s", nome1); //le no maximo 99 caracteres seguidos
scanf("%99s", nome2);  //ou ate encontar ' ', \t, \n

//"consumir" o \n anterior: utilize um espaco
//      |
scanf(" %99[^\n]", nome3);
scanf(" %99[^\n]", nome4);

printf("1 %s\n", nome1);
printf("2 %s\n", nome2);
printf("3 %s\n", nome3);
printf("4 %s\n", nome4);
```

Exemplo:

Entradas	**Saída:**
Joao	1 Joao
Maria	2 Maria
Jose da Silva	3 Jose da Silva
Antonio Souza	4 Antonio Souza

String - Sequência de caracteres - leituras seguidas

```c
char nome1[100], nome2[100], c;
int i;

for(i=0;i<4;i++) {
    scanf("%99s", nome1); //sequencia de palavras
    printf("%s\n", nome1);
}

//"pular" separadores(espacos, tab, \n) anteriores
for(i=0;i<4;i++) {
    //      |
    scanf(" %99[^\n]", nome2); //sequencia de frases
    printf("%s\n", nome2);
}

for(i=0;i<4;i++) {
    //      |
    scanf(" %c", &c);
    printf("%c\n", c);
}
```

String - Sequência de caracteres - percorrer

```c
char nome1[100];
scanf("%s", nome1); //sem & : vetor = endereco da 1a posicao

i = 0;
while(nome1[i]!= '\0') //percorrendo a string
{
    //especificador %c
    printf("%c", nome1[i]);
    i++;
}
printf("\n");
```

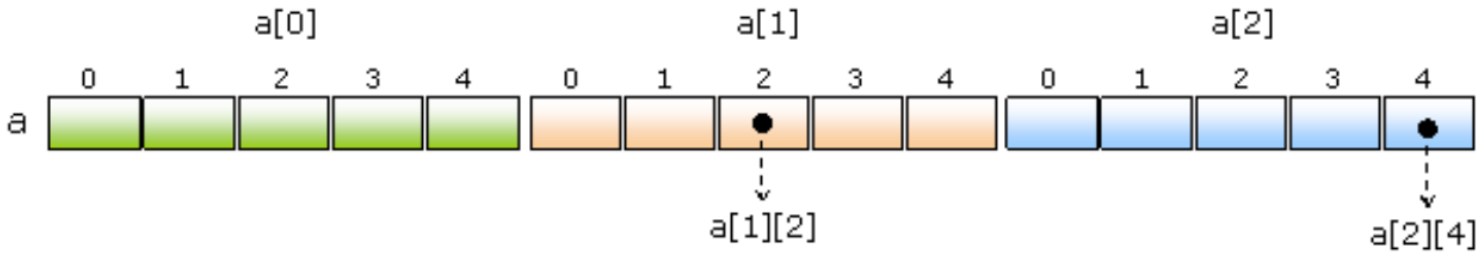
String - Sequência de caracteres - percorrer

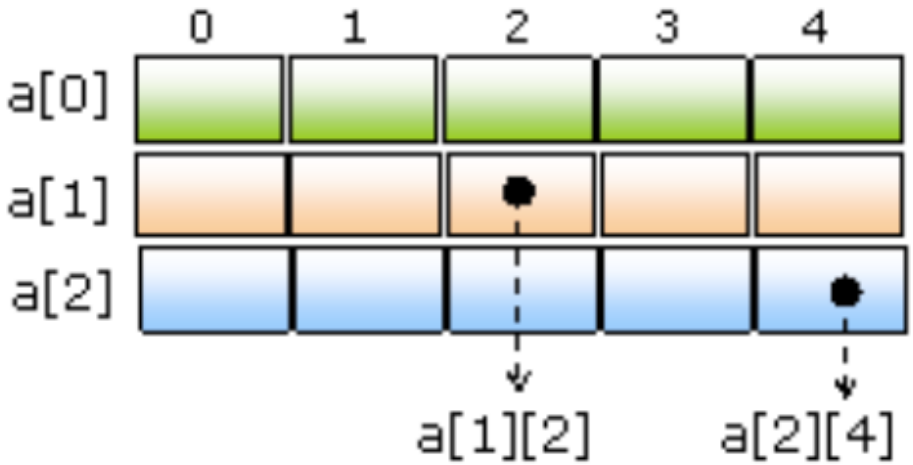
```c
#include <stdio.h>

//Qual a utlidade da função abaixo?
int string(char s1[], char s2[]) {
    int i=0;
    while(s1[i]!='\0' && s2[i]!='\0' && s1[i]==s2[i])
        i++;

    return s1[i] - s2[i];
}
```

Matriz ou Array multidimensional

- Acesso aleatório dada a posição da linha e coluna
- Alocação estática:
  - A variável aponta para o endereço da primeira posição
  - Alocação contígua

```
1 int  a[3][5];
```





Matriz ou Array multidimensional - declaração e inicialização

```
1  //Inicialização na declaração
2  int a[3][5]={{10,6,7,12,11},{23,32,14,52,22},{33,17,18,54,28}};
3  int a[][5]={{10,6,7,12,11},{23,32,14,52,22},{33,17,18,54,28}};
4  int a[][5]={10,6,7,12,11,23,32,14,52,22,33,17,18,54,28};
5
6  //Inicialização por atribuição
7  a[0][0] = 3;
8  a[1][4] = 12;
9  int b[2][2] = {}; //??
10
11 //Inicialização pela entrada padrão
12 int lin, col;
13 //para cada linha
14 for(lin=0; lin<3; lin++) {
15     //e cada coluna
16     for(col=0; col<5; col++) {
17         scanf("%d", &a[lin][col]);
18     }
19 }
```

	0	1	2	3	4
0	10	6	7	12	11
1	23	32	14	52	22
2	33	17	18	54	28

Matriz ou Array multidimensional - percorrer

```c
int a[3][5]={{10,6,7,12,11},{23,32,14,52,22},{33,17,18,54,28}};

//Impressão: posição por posição
int lin, col;
for(lin=0; lin<3; lin++)
{
    for(col=0; col<5; col++)
    {
        printf("%5d", a[lin][col]);
    }
    printf("\n");
}
```

Saída:
10 6 7 12 11
23 32 14 52 22
33 17 18 54 28

Matriz ou Array multidimensional - percorrer

```c
//passando matriz -> colunas obrigatorio
void matriz(int lin, int col, int a[lin][col]) {
    //Impressão: posição por posição
    int i, j;
    for(i=0; i<lin; i++)
    {
        for(j=0; j<col; j++)
        {
            printf("%5d", a[col][lin]);
        }
        printf("\n");
    }
}
```

```c
int v[3][5]={{10,6,7,12,11},
             {23,32,14,52,22},
             {33,17,18,54,28}};

matriz(3, 5, v);  \\saída?
```

Matriz ou Array multidimensional - percorrer

```c
//passando matriz -> colunas obrigatorio
void matriz(int lin, int col, int a[lin][col]) {
    //Impressão: posição por posição
    int i, j;
    for(i=0; i<lin; i++)
    {
        for(j=0; j<col; j++)
        {
            printf("%5d", a[col][lin]);
        }
        printf("\n");
    }
}
```

```c
int v[3][5]={{10,6,7,12,11},
             {23,32,14,52,22},
             {33,17,18,54,28}};

matriz(3, 5, v);  \\saída?
```

10 23 33
6 32 17
7 14 18
12 52 54
11 22 28

Matriz x String - declaração e inicialização

```c
int main()
{
    //Vetor de strings
    //inicialização na declaração
    char nomes[5][20] = {
        "Jose Silva",
        "Maria Silva",
        "Antonio dos Santos",
        "Pedro dos Santos",
        "Joao da Silva"};
    int i;

    for(i = 0; i < 5; i += 1)
        printf("%s\n", nomes[i]);
}
```

Saída:

```
Jose Silva
Maria Silva
Antonio dos Santos
Pedro dos Santos
Joao da Silva
```

Matriz x String - inicialização pela entrada

```c
int main()
{
    char nomes[5][20];

    //Vetor de strings
    //inicialização pela entrada padrão
    int i;
    for(i=0;i<5;i++) {
        //      |
        scanf(" %99[^\n]", nomes[i]);
    }

    for(i = 0; i < 5; i += 1) {
        printf("%s\n", nomes[i]);
    }
}
```

Roteiro

Estrutura de Dados Heterogêneos: struct (registro)

- Variável: armazena um conteúdo de um TIPO específico
  - `int`, `float`, `char`: representam um dado primitivo/elementar
  - `struct`: representa um conjunto variado de dados
    - Elementares
    - Homogêneos
    - Heterogêneos

```
1 int var1;
2 float var2;
3 char var3;
4
5 Data var4;    //Data é um tipo: o que armazenam?
6 Pessoa var5; //Pessoa é um tipo: o que armazenam?
```

Estrutura de Dados Heterogêneos: struct - definição

```c
//criar/definir o tipo
struct nome_da_estrutura {

    tipo_do_campo1 campo1; //cada elemento é denominado "campo"
    tipo_do_campo2 campo2;
    ...

}; //ponto e vírgula no final
```

```c
//Opcional: renomeando o tipo de "struct data" para "Data"
typedef struct data Data;

//definir o tipo Data
struct data {
    int dia, mes, ano;
};

//definir o tipo Pessoa
typedef struct {
    int cpf;
    Data nascimento;
    char nome[100];
}Pessoa;
```

Estrutura de Dados Heterogêneos: struct - declaração

```c
#include <stdio.h>
//definições
struct data {
    int dia, mes, ano;
};

typedef struct {
    int cpf;
    struct data nascimento; //tipo "struct data"
    char nome[100];
}Pessoa;

int main() {

    //declarações
    Pessoa diretor;       //variável "diretor"
    Pessoa alunos[1000]; //vetor "alunos"
    Pessoa vazia = {0, {0,0,0}, "nada"};

    ...

    return 0;
}
```

Estrutura de Dados Heterogêneos: struct - inicialização

```
1 //declaração
2 Pessoa diretor;       //variável "diretor"
3 Pessoa alunos[1000]; //vetor "alunos"
4
5 //inicialização pela entrada padrão (arquivo stdin)
6 scanf("%d", &diretor.cpf); //campos são acessado por '.'
7 scanf("%[^\n]", diretor.nome); //&?
8 scanf("%d%d%d", &diretor.nascimento.dia, &diretor.nascimento.
      mes, &diretor.nascimento.ano);
```

Estrutura de Dados Heterogêneos: struct - inicialização

```
1  //declaração
2  Pessoa diretor;       //variável "diretor"
3  Pessoa alunos[1000]; //vetor "alunos"
4
5  //inicialização pela entrada de arquivo
6  FILE *fp = fopen(fp, "discentes.txt");
7  if(fp){
8      for(int i=0; i<1000; i++){
9          fscanf(fp, "%d", &alunos[i].cpf);
10         fscanf(fp, "%[^\n]", alunos[i].nome); //&?
11         scanf("%d%d%d", &alunos[i].nascimento.dia, &alunos[i].
       nascimento.mes, &alunos[i].nascimento.ano);
12     }
13     fclose(fp);
14 }
15
16 //inicialização por cópia
17 Pessoa copia;
18 copia = diretor; //cópia do conteúdo da variável "diretor"
19
20 //alterando o campo "cpf" da variável "copia"
21 copia.cpf = 888888;
```

Estrutura de Dados Heterogêneos: struct - função

```c
struct endereco{ char rua[50]; int num; };

void imprimir_endereco(struct endereco); //parâmetro
struct endereco ler_endereco(); //retorno

int main() {
    struct endereco e;
    e = ler_endereco();
    imprimir_endereco(e);
    return 0;
}
//retornar struct
struct endereco ler_endereco() {
    struct endereco ender;
    scanf("%s", ender.rua);
    scanf("%d", &ender.num);
    return ender;
}
//por valor = somente conteudo
void imprimir_endereco(struct endereco ender) {
    printf("%s\n", ender.rua);
    printf("%d\n", ender.num);
    ender.num = 20; //nao altera a original
}
```