Linguagem C - Vetores e Registros

Prof. Antonio L. Apolinário Jr.

UFBA/IM/DCC/BCC 2015.2

Roteiro

- Vetores
- Registros
- Exercícios

Vetores e Matrizes

Vetores

- Agregados homogêneos de valores
- Associado a um bloco de memória
  - Endereço Base
  - Numero de elementos
  - Tipo do elemento

Vetores

- Declaração em C:

```
int V[20];

float M[10][10]

#define MAX 200
int P[MAX];
```

Vetores

- Acesso aos elementos de um vetor/matriz em C:

```
V[0] = 10;

M[1][7] = 20.4;

P[i+j*4]=40;
```

- Como calcular o acesso aos endereços de memória do vetor?
- Como verificar se esses endereços são válidos?

Vetores

- Como calcular o acesso aos endereços de memória do vetor?
  - **`&(V[i]) = &V+i*sizeof(int);`**
  - **`&(M[i][j]) = (&M+i*numCols+j)*sizeof(int);`**
- Como verificar se esses endereços são válidos?
  - Não há geração de código pelo compilador para essa verificação
  - Feita explicitamente pelo programador
    - Caso contrário: erro de segmentação (*segmentation fault*)

Vetores

Exemplo:

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <stdbool.h>
#define MAX 200

int main() {

int V[MAX];
bool OK;
int  i, num;
    srand ( time(NULL) );
    for (i=0; i < MAX; i++) {
     V[i] = rand() % 20;
     }
do {
 printf("Forneca um numero inteiro no inervalo [0,20[ :");
 scanf("%d", &num);
 printf("\n");
 } while ( (num < 0) && (num > 20));
    OK = true;
i = MAX-1;
while(OK) {
 OK = ((i > 0) && (V[i] != num));
 i--;
 }
if (i<0)
 printf("o numero %d nao pertence ao conjunto\n", num);
else
 printf("o numero %d pertence ao conjunto\n", num);
}
```

Registros

Registros

- Agregados heterogêneos de valores
- Associado a um bloco de memória
  - Endereço Base
  - Layout dos campos
  - Tipo de cada campo

Registros

- Declaração em C:

```
struct ponto { float x;
               float y;
               float z;
               int cor; };

struct ponto v;
```

- A variável v possui a estrutura de ponto, ou seja, todos os seus campos

Registros

- Declaração em C:

```
struct ponto { float x;
               float y;
               float z;
               int cor; };

struct ponto v;
```

- Acesso aos campos:

```
v.x = 10.2; v.y = v.z = 0.0;
v.cor = 255;
```

Registros

- É possível declarar um tipo definido pelo usuário associado ao registro

```
typedef struct ponto { float x;
                       float y;
                       float z;
                       int cor; } tPonto;

tPonto v;
```

- Acesso aos campos continua como antes:

```
v.x = 10.2; v.y = v.z = 0.0;
v.cor = 255;
```

Registros

- Exemplo:

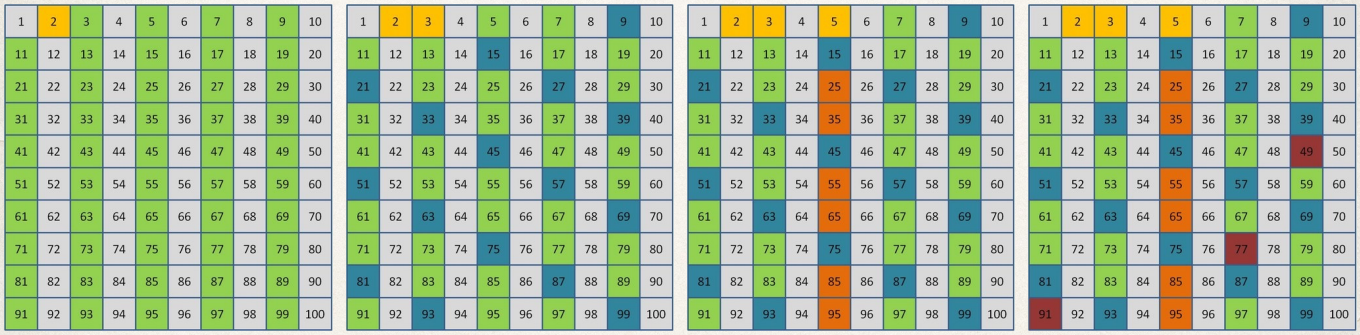
```
/* Tipos Abstratos e Registros */
#include<stdio.h>
typedef struct data    { int dia;
                         int mes;
                         int ano;
                       } tData;
typedef struct periodo {
                    tData inicio;
                    tData fim;
                    int numDias;
                    } tPeriodo;
tPeriodo p = { { 1, 1, 2012},
               { 1, 1, 2013},
               365 };
struct periodo q;
```

```
int main() {
 printf("Tipos Abstratos e Registros!!\n");
 printf("Periodo p: inicio = %d/%d/%d\ninicio =
         %d/%d/%d\ndias=%d\n",
      p.inicio.dia, p.inicio.mes, p.inicio.ano,
      p.fim.dia, p.fim.mes, p.fim.ano,
      p.numDias );
 q.inicio.dia = p.fim.dia;
 q.inicio.mes = p.fim.mes;
 q.inicio.ano = p.fim.ano;
 q.fim.dia = p.inicio.dia;
 q.fim.mes = p.inicio.mes;
 q.fim.ano = p.inicio.ano;
 q.numDias = -365;
 printf("Periodo q: inicio = %d/%d/%d\ninicio =
       %d/%d/%d\ndias=%d\n",
      q.inicio.dia, q.inicio.mes, q.inicio.ano,
      q.fim.dia,q.fim.mes,q.fim.ano,q.numDias);
 return 0; }
```

Exercício

Exercício

1. Faça um programa em linguagem C que leia um número inteiro positivo **n** e indique todos os números primos menores que **n**. Utilize para isso o Algoritmo Crivo de Eratóstenes.
   Exemplo: para n=100, 4 primeiras iterações do algoritmo



Exercício

2. Faça um programa em C que gere um conjunto aleatório de pontos tridimensionais, dentro do intervalo de [-100, 100]. No máximo serão gerados 200 pontos, sendo que a quantidade exata deve ser fornecida pelo usuário.
   A cada ponto deve ser associada uma cor, representada no espaço de cores RGB, dentro do intervalo de [0,255]. As cores R,G e B devem ser geradas a partir das coordenadas (x,y,z) dos pontos.
   Após a geração da nuvem de pontos, seu programa deve calcular:
   1. Centro de massa da nuvem;
   2. A cor média dos pontos;
   3. Quais pontos e cores se repetem e quantas vezes isso ocorre.