Agenda

1. Introdução
2. Tipo estrutura
3. Definição de “novos” tipos
4. Aninhamento de estruturas
5. Vetores de estruturas
6. Vetores de ponteiros para estruturas

Introdução

Introdução

- Representação dos dados de maneira abstrata
  - Exemplo: pontos no espaço bidimensional (x, y).

Introdução

- Representação dos dados de maneira abstrata
  - Exemplo: cadastro dos alunos matriculados em determinada disciplina.

Introdução

- A linguagem C oferece mecanismos para estruturar dados nos quais as informações são compostas por diversos campos.

Tipo estrutura

Tipo estrutura

- Uma estrutura, em C, serve basicamente para agrupar diversas variáveis dentro de um único contexto.
  - Exemplo: estrutura **ponto**.

```
struct ponto {
    float x;
    float y;
};
```

Fonte: CELES; CERQUEIRA; RANGEL, 2004.

Tipo estrutura

- A estrutura **ponto** representa um tipo.

```
struct ponto p;

p.x = 10.0;
p.y = 5.0;
```

Fonte: CELES; CERQUEIRA; RANGEL, 2004.

Tipo estrutura

- Exemplo: captura e impressão das coordenadas de um ponto qualquer.

```c
#include <stdio.h>

struct ponto {
    float x;
    float y;
};

int main (void) {
    struct ponto p;
    printf("Digite as coordenadas do ponto (x y):");
    scanf("%f %f", &p.x, &p.y);
    printf("O ponto fornecido foi: (%.2f, %.2f)\n",
                                        p.x, p.y);
    return 0;
}
```

Fonte: CELES; CERQUEIRA; RANGEL, 2004.

Tipo estrutura

- Ponteiro para estruturas

```
struct ponto *pp;

(*pp).x = 12.0;
pp->x = 12.0;
```

Fonte: CELES; CERQUEIRA; RANGEL, 2004.

Tipo estrutura

- Passagem de estruturas para funções

```
void imprime (struct ponto p) {
    printf("O ponto fornecido foi: (%.2f, %.2f)\n",
                                        p.x, p.y);
}
```

  - Problemas com esta abordagem:
    - Não há como alterar os valores dos elementos da estrutura original;
    - Copiar uma estrutura inteira para a pilha pode ser uma operação custosa.

Tipo estrutura

- Passagem de estruturas para funções

```
void imprime (struct ponto *pp) {
    printf("O ponto fornecido foi: (%.2f, %.2f)\n",
                                        pp->x, pp->y);
}

void captura (struct ponto *pp) {
    printf("Digite as coordenadas do ponto (x y):");
    scanf("%f %f", &pp->x, &pp->y);
}
```

Fonte: CELES; CERQUEIRA; RANGEL, 2004.

Tipo estrutura

- Passagem de estruturas para funções

```
int main (void) {
    struct ponto p;
    captura(&p);
    imprime(&p);
    return 0;
}
```

Fonte: CELES; CERQUEIRA; RANGEL, 2004.

Tipo estrutura

- Alocação dinâmica de estruturas

```
struct ponto* p;
p = (struct ponto*) malloc (sizeof(struct ponto));
...
p->x = 12.0;
...
```

Fonte: CELES; CERQUEIRA; RANGEL, 2004.

Definição de “novos” tipos

Definição de “novos” tipos

- A linguagem C permite criar nomes de tipos.

```
typedef float Real;

typedef unsigned char Uchar;
typedef int* Pint;
typedef float Vetor[4];

Vetor v;
...
v[0] = 3.0;
...
```

Fonte: CELES; CERQUEIRA; RANGEL, 2004.

Definição de “novos” tipos

- Em geral, definimos nomes de tipos para estruturas.

```
struct ponto {
        float x;
        float y;
};
typedef struct ponto Ponto;

typedef struct ponto *PPonto;

//usando o mesmo typedef
typedef struct ponto Ponto, *PPonto;
```

Fonte: CELES; CERQUEIRA; RANGEL, 2004.

Definição de “novos” tipos

- Em geral, definimos nomes de tipos para estruturas.

```
typedef struct ponto {
        float x;
        float y;
} Ponto;
```

Fonte: CELES; CERQUEIRA; RANGEL, 2004.

Aninhamento de estruturas

Aninhamento de estruturas

- Os campos de uma estrutura podem ser outras estruturas previamente definidas.
- Exemplo:
  - Estrutura **ponto** e função **distancia**;
  - Estrutura **círculo** e função **interior**.

Aninhamento de estruturas

- Exemplo:

```
struct ponto {
    float x;
    float y;
};
typedef struct ponto Ponto;
```

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

```
float distancia (Ponto* p, Ponto* q) {
    float d = sqrt((q->x - p->x)*(q->x - p->x)
                 + (q->y - p->y)*(q->y - p->y));
    return d;
}
```

Fonte: CELES; CERQUEIRA; RANGEL, 2004.

Aninhamento de estruturas

- Exemplo:

```
struct circulo {
    float x, y;/*centro do círculo*/
    float r;   /*raio do círculo*/
};
```

Fonte: CELES; CERQUEIRA; RANGEL, 2004.

Aninhamento de estruturas

- Exemplo:

```
struct circulo {
    Ponto p;/*centro do círculo*/
    float r;/*raio do círculo*/
};
typedef struct circulo Circulo;
```

```
int interior (Circulo* c, Ponto* p) {
    float d = distancia(&c->p, p);
    return (d < c->r);
}
```

Fonte: CELES; CERQUEIRA; RANGEL, 2004.

Vetores de estruturas

Vetores de estruturas

- Exemplo: cálculo do centro geométrico.

$$x = \frac{\sum x_i}{n} \qquad \mathrm{y} = \frac{\sum y_i}{n}$$

Vetores de estruturas

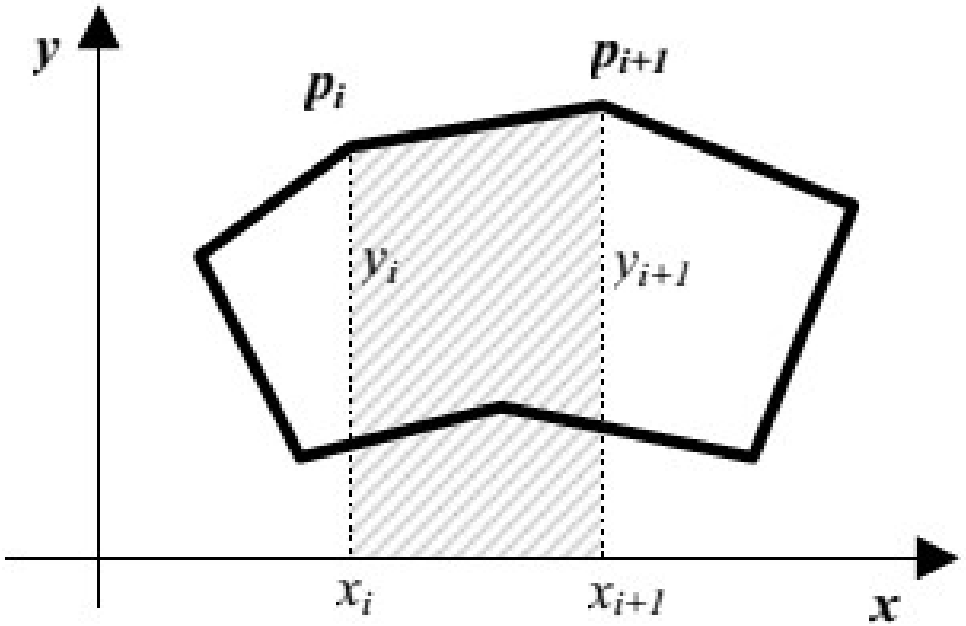
- Exemplo: cálculo do centro geométrico.

```
Ponto centro_geom (int n, Ponto* v) {
    int i;
    /*declara e inicializa ponto*/
    Ponto p = {0.0f, 0.0f};
    for (i = 0; i < n; i++) {
        p.x += v[i].x;
        p.y += v[i].y;
    }
    p.x /= n;
    p.y /= n;
    return p;
}
```

Fonte: CELES; CERQUEIRA; RANGEL, 2004.

Vetores de estruturas

- Exemplo: cálculo da área de um polígono (n pontos).

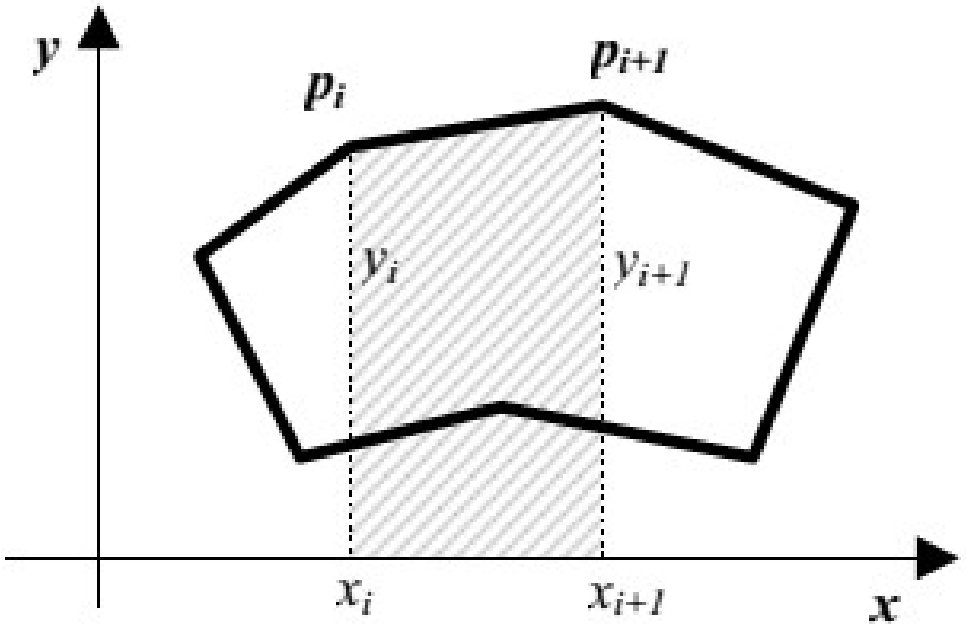


Fonte: CELES; CERQUEIRA; RANGEL, 2004.

  - A área pode ser calculada pela soma das áreas dos trapézios formados pelos lados do polígono e o eixo x.

Vetores de estruturas

- Exemplo: cálculo da área de um polígono (n pontos).



Fonte: CELES; CERQUEIRA; RANGEL, 2004.

  - Área do trapézio definido pela aresta que vai do ponto p_i ao ponto p_{i+1}:
    - $a = (x_{i+1} - x_i)(y_{i+1} + y_i)/2$

Vetores de estruturas

- Exemplo: cálculo da área de um polígono (n pontos).

```
float area (int n, Ponto* p) {
    int i, j;
    float a = 0;
    for (i = 0; i < n; i++) {
        /*próximo índice (incremento circular)*/
        j = (i + 1) % n;
        a += (p[j].x - p[i].x)*(p[i].y + p[j].y)/2;
    }
    return fabs(a);
}
```

Fonte: CELES; CERQUEIRA; RANGEL, 2004.

Vetores de estruturas

- Exemplo: cálculo da área de um polígono (n pontos).

```
int main (void) {
    Ponto p[3] = {{1.0,1.0},{5.0,1.0},{4.0,3.0}};
    printf("area = %f\n", area(3, p));
    return 0;
}
```

Fonte: CELES; CERQUEIRA; RANGEL, 2004.

Vetores de estruturas

- Exercício: cálculo da área de um polígono (n pontos)
  - (CELES; CERQUEIRA; RANGEL, 2004): Alterar o programa ilustrado para capturar do teclado o número de pontos que delimitam o polígono. O programa então alocaria dinamicamente o vetor de pontos, capturaria as coordenadas dos pontos e, chamando a função **area**, exibiria o valor da área.

Vetores de ponteiros para estruturas

Vetores de ponteiros para estruturas

- O uso de vetores de ponteiros é útil quando temos de tratar um conjunto de elementos complexos.
- Exemplo: armazenar uma tabela com dados de alunos
  - matrícula: número inteiro;
  - nome: cadeia com até 80 caracteres;
  - endereço: cadeia com até 120 caracteres;
  - telefone: cadeia com até 20 caracteres.

Vetores de ponteiros para estruturas

- Estrutura **aluno** (primeira opção – problema?):

```
struct aluno {
    int mat;
    char nome[81];
    char end[121];
    char tel[21];
};
typedef struct aluno Aluno;
```

```
#define MAX 100
Aluno tab[MAX];
...
tab[i].mat = 9912222;
...
```

Fonte: CELES; CERQUEIRA; RANGEL, 2004.

Vetores de ponteiros para estruturas

- Estrutura **aluno** (opção mais adequada):

```
struct aluno {
    int mat;
    char nome[81];
    char end[121];
    char tel[21];
};
typedef struct aluno Aluno;
```

```
#define MAX 100
Aluno* tab[MAX];
```

Fonte: CELES; CERQUEIRA; RANGEL, 2004.

Vetores de ponteiros para estruturas

- Implementação de algumas funcionalidades:

```
void inicializa (int n, Aluno** tab) {
    int i;
    for (i = 0; i < n; i++)
        tab[i] = NULL;
}
```

Fonte: CELES; CERQUEIRA; RANGEL, 2004.

Vetores de ponteiros para estruturas

- Implementação de algumas funcionalidades:

```
void preenche (int n, Aluno** tab, int i) {
    if (i < 0 || i >= n) {
        printf("Indice fora do limite do vetor.\n");
        exit(1);/*aborta o programa*/
    }
    if (tab[i] == NULL)
        tab[i] = (Aluno*) malloc(sizeof(Aluno));

    printf("Entre com a matricula: ");
    scanf("%d", &tab[i]->mat);
    printf("Entre com o nome: ");
    scanf(" %80[^\n]", tab[i]->nome);
    printf("Entre com o endereco: ");
    scanf(" %120[^\n]", tab[i]->end);
    printf("Entre com o telefone: ");
    scanf(" %20[^\n]", tab[i]->tel);
}
```

Fonte: CELES; CERQUEIRA; RANGEL, 2004.

Vetores de ponteiros para estruturas

- Implementação de algumas funcionalidades:

```
void retira (int n, Aluno** tab, int i) {
    if (i < 0 || i >= n) {
        printf("Indice fora do limite do vetor.\n");
        exit(1);/*aborta o programa*/
    }
    if (tab[i] != NULL) {
        free(tab[i]);
        tab[i] = NULL;/*indica que na posição
                        não mais existe dado*/
    }
}
```

Fonte: CELES; CERQUEIRA; RANGEL, 2004.

Vetores de ponteiros para estruturas

- Implementação de algumas funcionalidades:

```
void imprime (int n, Aluno** tab, int i) {
    if (i < 0 || i >= n) {
        printf("Indice fora do limite do vetor.\n");
        exit(1);/*aborta o programa*/
    }
    if (tab[i] != NULL){
        printf("Matricula: %d\n", tab[i]->mat);
        printf("Nome: %s\n", tab[i]->nome);
        printf("Endereco: %s\n", tab[i]->end);
        printf("Telefone: %s\n", tab[i]->tel);
    }
}
```

Fonte: CELES; CERQUEIRA; RANGEL, 2004.

Vetores de ponteiros para estruturas

- Implementação de algumas funcionalidades:

```
void imprime_tudo (int n, Aluno** tab) {
    int i;
    for(i = 0; i < n; i++)
        imprime(n, tab, i);
}
```

Fonte: CELES; CERQUEIRA; RANGEL, 2004.

Vetores de ponteiros para estruturas

- Implementação de algumas funcionalidades:

```
int main (void) {
    Aluno* tab[10];
    inicializa(10, tab);
    preenche(10, tab, 0);
    preenche(10, tab, 1);
    preenche(10, tab, 2);
    imprime_tudo(10, tab);
    retira(10, tab, 0);
    retira(10, tab, 1);
    retira(10, tab, 2);
    return 0;
}
```

Fonte: CELES; CERQUEIRA; RANGEL, 2004.

Referências

- CELES, W.; CERQUEIRA, R.; RANGEL, J. L. **Introdução a Estruturas de Dados: com técnicas de programação em C.** Rio de Janeiro: Elsevier, 2004.
- OLIVEIRA, U. **Programando em C** – Volume I – Fundamentos. Ciência Moderna, 2008.