Algoritmos e estruturas de Dados A04

Estruturas

Prof. Mauri Ferrandin

Estruturas

Cadastramento dos dados de uma pessoa

```c
#include <stdio.h>

int main()
{
        char nome[50];
        char end[50];
        int tel;
        int matricula;


        // ... continuação


}
```

Estruturas

Cadastramento dos dados de 4 pessoas

```
#include <stdio.h>

int main()
{
  char nome1[50],nome2[50],nome3[50],
       nome4[50];
  char end1[50], end2[50], end3[50],
       end4[50];
  int tel1, tel2, tel3, tel4;
  int matricula1, matricula2, matricula3,
       matricula4;

  // ... continuação

}
```

Estruturas

Estrutura (struct em C) é um tipo estruturado de dados

- permite que agrupar variáveis de vários tipos sob o mesmo nome (identificador);
- Esse novo identificador passará a ser um novo tipo de dados (tal como o int ou float.) construído a partir de outros tipos

```
struct <identificador> {
        <tipo> campo_um;
        <tipo> campo_dois;
};
```

Estruturas

Exemplo

```
...
struct dadosfunc{
        char nome[50];
        char end[50];
        int tel;
        int matricula;
};
...
```

- Uma estrutura (dadosfunc)
- Quatro campos
  - Nome
  - End
  - Tel
  - Matricula

Estruturas

Exemplo

```
...
struct dadosprod{
        float peso;
        float preco;
};
...
```

- Uma estrutura (dadosprod)
- Dois campos
  - peso
  - preco

Estruturas

- Declaração das instâncias com a estrutura:

```
struct <identificador> {
        <tipo> campo_um;
        <tipo> campo_dois;
}inst1, inst2, inst3;
```

```
...
struct produto{
        int peso;
        float preco;
 } maca, banana, melao;
…
```

Estruturas

- Declaração como variável normal:

```
struct <identificador> <nome_variavel>;
```

```
struct dadosfunc{
...
}

int main()
{

    struct dadosfunc func1;
    struct dadosfunc func2;
    struct dadosfunc funcionarios[10];
    ...

}
```

Estruturas

Inicialização de estruturas no programa:

- Listas de inicializadores

```
struct <identificador> <nome> =
                    {val1, val2,..., valN};
```

```c
#include <stdio.h>

struct produto{
    int peso;
    float preco;
};

void main(){
    // inicializando todos os campos
    struct produto uva = {55, 3.77};
    // inicializando campo a campo
    struct produto milho;
    milho.peso = 5;
    milho.preco = 5.4;
    // copiando de uma struct para outra
    struct produto milho2 = milho;
}
```

Estruturas

- Acesso aos campos

```
<nome instancia>.<campo>
```

```
struct dadosfunc{
        char nome[50];
        char end[50];
        int tel;
        int matricula;
};

int main(){
   ...
   struct dadosfunc p1;
   p1.matricula=171;
   strcpy(p1.nome,"Joao Paulo");
   scanf("%s", &p1.end);
   printf("%s", p1.end);
   ...
}
```

Estruturas

Exemplo

- Utilização da estrutura

```c
#include <stdio.h>

struct data {
        int ano;
        int mes;
        int dia;
};

int main() {
        struct data hoje;
        hoje.ano = 2014;
        hoje.mes = 8;
        hoje.dia = 18;
        printf("Hoje e %d/%d/%d.\n",
          hoje.ano, hoje.mes, hoje.dia );
}
```

Estruturas

Exemplo

- 2 estruturas
- campos com o mesmo nome

```
#include <stdio.h>

struct point2d{
        int x, y;
};

struct point3d{
        int x, y, z ;
};

int main() {
        struct point2d p1;
        struct point3d p2;

        p1.x = 50;
        p2.x = 37;
        printf("P1.x = %d \nP2.x = %d", p1.x,
          p2.x);
}
```

Criação de tipo de dados

- A palavra chave **typedef** permite a definição de um novo tipo de dado
- typedef → **type def**inition

```
...
typedef int inteiro;
inteiro num3 = 4;
...
```

- Extremamente útil para dar nome a tipos complexos, como as estruturas

```
...
typedef struct{
        int x, y;
}Point2d;
...
```

Criação de tipo de dados

- Exemplo sem typedef

```c
#include <stdio.h>

struct point2d{
  int x, y;
};

struct point3d{
  int x, y, z ;
};

int main(){
  struct point2d p1;
  struct point3d p2;

  p1.x = 55;
  p2.x = 37;
}
```

- Exemplo com typedef

```c
#include <stdio.h>

typedef struct{
  int x, y;
}Point2d;

typedef struct{
  int x, y, z;
}Point3d;

int main(){
  Point2d p1;
  Point3d p2;

  p1.x = 55;
  p2.x = 37;
}
```

Struct como parâmetro para função

- Exemplo sem typedef

```c
#include <stdio.h>

struct point2d{
  int x, y;
};

void mostra(struct point2d p){
  printf("x = %d, y = %d\n",
p.x, p.y);
}

int main(){
  struct point2d p1;
  p1.x = 55;
  p1.y = 37;
  mostra(p1);
}
```

- Exemplo com typedef

```c
#include <stdio.h>

typedef struct{
  int x, y;
}Point2d;

void mostra(Point2d p){
  printf("x = %d, y = %d\n",
p.x, p.y);
}

int main(){
  Point2d p1;
  p1.x = 55;
  p1.y = 37;
  mostra(p1);
}
```

Aninhamento de estruturas

Uma struct usada para definir outra struct

```c
#include <stdio.h>

struct Endereco{
  char rua[50];
  int numero;
};

struct Pessoa {
  int codigo;
  char nome[50];
  int idade;
  struct Endereco end;
};

// continua ao lado
```

```c
// continuacao

int main(){

  struct Pessoa p1;
  p1.codigo = 1022;
  strcpy(p1.nome,"Joao");
  p1.idade = 55;
  strcpy(p1.end.rua,"Rua 33");
  p1.end.numero = 5351;

  printf("%s\n", p1.nome);
  printf("%s\n", p1.end.rua);

}
```

Vetor de estruturas

Exemplo com struct aninhada e vetor

```c
#include <stdio.h>

struct Endereco{
  char rua[50];
  int numero;
};

struct Pessoa {
  int codigo;
  char nome[50];
  int idade;
  struct Endereco end;
};

// continua ao lado
```

```c
// continuacao
int main(){
  struct Pessoa p[5];

  p[0].codigo = 12;
  strcpy(p[0].nome, "José");
  p[0].idade = 25;
  strcpy(p[0].end.rua,"Rua2");
  p[0].end.numero = 222;

  p[3].codigo = 1022;
  strcpy(p[3].nome, "Joao");
  p[3].idade = 55;
  strcpy(p[3].end.rua,"Rua2");
  p[3].end.numero = 5351;
}
```

Vetor de estruturas

Exemplo com struct aninhada definida como tipo e vetor

```c
#include <stdio.h>

typedef struct{
  char rua[50];
  int numero;
}Endereco;

typedef struct{
  int codigo;
  char nome[50];
  int idade;
  Endereco end;
}Pessoa ;

// continua ao lado
```

```c
// continuacao
int main(){
  Pessoa p[5];

  p[0].codigo = 12;
  strcpy(p[0].nome, "José");
  p[0].idade = 25;
  strcpy(p[0].end.rua,"Rua2");
  p[0].end.numero = 222;

  p[3].codigo = 1022;
  strcpy(p[3].nome, "Joao");
  p[3].idade = 55;
  strcpy(p[3].end.rua,"Rua2");
  p[3].end.numero = 5351;
}
```