UNESPAR
Universidade Estadual do Paraná

Algoritmos e Técnicas de Programação

Aula 5 - Registros

Prof. Guilherme Guerino

Contextualização

- **Problema da aula de Arranjos:**
  - Professor precisava armazenar as notas dos 30 alunos. Para armazenar todas as notas, foi definido um arranjo, no qual cada elemento era uma nota. Nos arranjos, todos os elementos são *homogêneos*, ou seja, têm o *mesmo tipo*;
  - Se o professor quiser armazenar, junto às notas de cada aluno, mais *algumas informações*, como o nome do aluno (uma string), seu número de faltas (valor inteiro) e o conceito final (caractere);
  - Essas informações não poderão ser armazenadas no arranjo das notas por serem de *tipos diferentes do tipo do arranjo*.

Introdução

- Registros são **estruturas** que permitem o armazenamento de **diferentes tipos de dados**.

- Além do **armazenamento**, é possível criar **novos tipos de dados**.
  - Os novos tipos são resultantes do agrupamento dos *tipos primitivos*, fornecidos pelas linguagens de programação.

Introdução

- Um registro também pode ser entendido como um **agrupamento de campos**.
  - Os registros também são conhecidos como *variáveis compostas heterogêneas*.

- Cada campo possui um tipo, que pode ser primitivo ou representar outro registro.

Declaração de Registros

- Na linguagem C, um registro é declarado por meio de uma **struct**.

```
struct nome_do_registro{
    tipo campo1;
    tipo campo2;
    ...
    tipo campoN;
};
```

Declaração de Registros

- Depois que o registro é declarado, pode-se considerar a existência de **um novo tipo de dado**, chamado *nome_do_registro*.

- O novo tipo é capaz de armazenar **diferentes informações**, associadas com os **diferentes campos**.

Exemplo

```
struct conta{
   char titular[30];
   int numero;
   float saldo;
};
```

Exemplo

- No exemplo apresentado, um registro chamado **conta** foi declarado.
- Uma variável do **tipo conta** armazena o nome do **titular**, o **numero** e o **saldo** da conta.
- Após a declaração do registro, é possível **declarar** e **utilizar** uma ou mais variáveis do **tipo conta**.

Declaração de Registros

- Um **registro** pode ser utilizado somente **dentro do bloco** onde foi declarado.

- Por exemplo, se um registro é definido **dentro do bloco main,** ele poderá ser utilizado somente por variáveis que **também estejam nesse bloco**.

Declaração de Registros

- Para evitar essa restrição, é importante que o registro seja **declarado fora do bloco main**.

- De preferência, a declaração de um registro deve acontecer logo após a **declaração das includes.**

Declaração de Registros

```
#include <stdio.h>
struct produto{
   int codigo;
   char descricao[50];
};
```

Declaração de Variáveis

- Para que um programa possa utilizar um registro, é necessário que pelo menos **uma variável do tipo do registro** seja declarada.

```
struct nome_do_registro nome_da_variável;
```

Exemplos

```
struct conta conta_familiar;

struct conta conta_universitaria;

struct conta conta_individual;
```

Exemplos

- O exemplo apresenta a declaração das variáveis **conta_familiar**, **conta_universitaria** e **conta_individual**.

- Cada variável é do tipo conta e pode armazenar três valores: **titular**, **numero** e **saldo**.

Novo tipo de dado

- Um **registro criado** representa um **novo tipo de dado**.
- **Todas as operações possíveis** com os tipos primitivos também podem ser realizadas com um novo tipo criado.

Novo tipo de dado

- Na aulas anteriores, foi observada a declaração de **variáveis simples**, **vetores** e **matrizes** de tipos primitivos.

- Uma vez que um registro foi declarado, pode-se declarar **vetores e matrizes de registro**.

Novo tipo de dado

```
struct conta m[10][6];

struct conta v[10];
```

Novo tipo de dado

- No exemplo apresentado, as variáveis **m** e **v** representam uma **matriz** e um **vetor** de registro, respectivamente.

- Cada **célula** da matriz ou cada **posição** do vetor armazena uma variável do **tipo struct conta**.

Acesso ao campo do registro

- Após a declaração de uma **variável do tipo registro**, tem-se que **cada campo deve ser acessado individualmente**.

- Por exemplo, após a declaração de uma variável do **tipo conta**, tem-se que os campos **titular**, **numero** e **saldo** devem ser **acessados individualmente**.

Acesso ao campo do registro

- A necessidade de acesso individual acontece na **manipulação de cada campo** do registro.

- É preciso informar o **nome da variável** e o nome do **campo desejado**, separados por **um ponto**.

Acesso ao campo do registro

- Para armazenar o valor 1 como código e a palavra geladeira como descrição, deve-se:

```
struct produto p;
p.codigo = 1;
strcpy(p.descricao,"geladeira");
```

Exemplo 1

```
#include <stdio.h>

struct produto{
   int codigo;
   char descricao[50];
};
.....
```

Exemplo 1

```
int main(){
    struct produto p;
    printf("\n Codigo: \n");
    scanf("%d", &p.codigo);
    fflush(stdin);
    printf("\n Descricao: \n");
    gets(p.descricao);
}
```

Exemplo 2

```
struct ponto{
   int eixo_x;
   int eixo_y;
};
...
```

Exemplo 2

```
struct ponto matriz[2][2];
matriz[0][0].eixo_x = 3;
matriz[0][0].eixo_y = 4;
```

Definição de nomes de tipos

- A linguagem C permite **criar nomes para os tipos de dados**.

- Com isso, **os nomes dos tipos das variáveis** podem ser renomeados.

Definição de nomes de tipos

- A palavra reservada **typedef** possibilita **renomear** um **determinado tipo** com **outro nome**.

- Isso é válido para **variáveis simples**, bem como para **vetores**, **matrizes** e **registros**...

Definição de nomes de tipos

```
typedef struct conta Conta;

typedef struct ponto Ponto;
```

Definição de nomes de tipos

- Após a utilização do **typedef**, o novo nome pode ser utilizado para se referir ao tipo.

Definição de nomes de tipos

```
Conta conta;
conta.numero = 1;
conta.titular = “cliente”;
Conta.saldo = 300.50;
```

Definição de nomes de tipos

```
Ponto ponto;
ponto.eixo_x = 6;
ponto.eixo_y = 6;
```

Exemplo

```
#include <stdio.h>
struct produto{
   int codigo;
   char descricao[50];
};
typedef struct produto Produto;
.....
```

Exemplo

```
int main(){
    Produto p;
    printf("\n Codigo: \n");
    scanf("%d", &p.codigo);
    fflush(stdin);
    printf("\n Descricao: \n");
    gets(p.descricao);
}
```

Dúvidas?