UNIFG

LAUREATE INTERNATIONAL UNIVERSITIES®

TÉCNICAS DE PROGRAMAÇÃO

Aula 09.1 – Manipulação de variáveis do tipo registro

Objetivos de Aprendizagem

1. Identificar a estrutura de um registro;
2. Identificar a criação do registro e como acessar seus membros;
3. Identificar os princípios básicos das formas de construção de programas para inserção e exclusão em um registro;
4. Identificar os princípios básicos das formas de construção de programas para pesquisa e alteração em um registro;
5. Desenvolver programas com registro.

Registros

- Nós já sabemos que um conjunto homogêneo de dados é composto por variáveis do mesmo tipo (vetores).
- Mas, e se tivermos um conjunto em que os elementos não são do mesmo tipo? Teremos, então, um conjunto **heterogêneo de dados, que são chamados de registros.**
- O registro é uma das principais formas de estruturar os dados no programa.
- Visa facilitar o agrupamento de variáveis de tipos diferentes, mas que possuem uma relação lógica.

Registros

- Um registro é um conjunto de uma ou mais variáveis, que podem ser de tipos diferentes, agrupadas sobre um único nome.
- O fato de variáveis agrupadas em um registro poderem ser referenciadas por um único nome, facilita a manipulação dos dados armazenados nestas estruturas.

Exemplo

- Imaginem uma estrutura que armazene as diversas informações do boletim de um aluno.
- O boletim é formado por um conjunto de informações logicamente relacionadas, porém de tipos diferentes, tais como:
  - número de matrícula (inteiro),
  - nome do aluno (caractere),
  - nome da disciplina (caractere),
  - média (real)
  - situação (caractere)
- Que são subdivisões do registro (elementos de conjunto), também chamadas de **campos.** Logo, um **registro** é composto por campos que são partes que especificam cada uma das informações.

Boletim de Notas

```
Matricula...: 12345
Nome.......: Michel
Disciplina..: Matemática
Média.......: 10.0
Situação....: Aprovado
```

Notem que o boletim é composto por informações de diferentes tipos. No entanto, todas as informações do boletim estão relacionadas ao mesmo aluno.
O agrupamento de informações de tipos diferentes, que tem uma relação lógica, facilitará a manipulação de dados.

Registros do tipo structs

- Um registro é um tipo de dado definido pelo usuário que agrupa variáveis relacionadas de diferentes tipos de dados.
- Uma declaração de registro inclui a **palavra-chave struct**, um **nome de estrutura** para referenciar a estrutura e **chaves** {} com **uma lista de** declarações de variáveis chamadas **membros**.

Exemplo de um registro

- Por exemplo:

```
struct curso {
  int id;
  char titulo[40];
  float carga_horaria;
};
```

- Esta instrução **struct** define um novo tipo de dados chamado **curso** que possui três membros.

Registros do tipo struct

- Os membros do registro podem ser de qualquer tipo de dado, incluindo tipos básicos, strings, vetores, ponteiros e até outros registros, como você aprenderá a seguir.
- Não esqueça de colocar um ponto e vírgula após a declaração da estrutura.
- Uma estrutura também é chamada de tipo de dados composto ou agregado.
- Alguns idiomas se referem às estruturas como registros.

Declarações usando struct

- Para declarar variáveis de um tipo de registro, use a palavra-chave **struct** seguida pela marcação **struct** e, por fim, o nome da variável.

Exemplo de struct

- Por exemplo, as instruções a seguir declaram um tipo de registro e, em seguida, usam a estrutura do aluno para declarar as variáveis a1 e a2:

```
struct aluno {
  int idade;
  float nota;
  char nome[40];
};

/* declara duas variáveis do tipo aluno */
struct aluno a1;
struct aluno a2;
a2.idade = 18;
a2.nota = 9.8;
a2.nome = "Maria";
printf("Nome do aluno %s", a2.nome); // Nome do aluno Maria
```

Declaração usando struct

- Uma variável **struct** é armazenada em um bloco continuo de memória.
- O operador **sizeof** deve ser usado para obter o número de bytes necessários para uma estrutura, assim como nos tipos de dados básicos.
- Uma variável struct também pode ser inicializada na declaração listando valores iniciais em ordem dentro de chaves:

```
struct aluno a1 = {19, 9, "João"};
struct aluno a2 = {22, 10, "Batman"};
```

Declaração usando struct

- Se você deseja inicializar uma estrutura usando chaves depois da declaração, também precisará digitar o cast (a conversão), como nas instruções:

```
struct aluno a1;
a1 = (struct aluno) {19, 9, "João"};

struct aluno a2 = {.nota = 8, .idade = 18,
.nome = "Maria"};
```

Inicialização de uma struct

- Você pode usar a inicialização de membro nomeado ao inicializar uma estrutura para inicializar os membros correspondentes:

```
struct aluno a1 = { .nota = 9, .idade =
 19, .nome = “João"};
```

- No exemplo acima, `.nota` refere-se ao membro da classe da struct.
- Da mesma forma, `.idade` e `.nome` se referem aos membros de idade e nome.

Acessando membros da estrutura

- Você acessa os membros de uma variável struct usando o "." (operador de ponto) entre o nome da variável e o nome do membro.
- Por exemplo, para atribuir um valor ao membro de idade da variável struct a1, use uma instrução como:

```
a1.idade = 19;
```

Acessando membros da estrutura

- Você também pode atribuir uma estrutura a outra do mesmo tipo:

```
struct aluno a1 = {19, 9, "Jackson"};
struct aluno a2;
//....
a2 = a1;
printf("Nome: %s", a2.nome);//Nome: Jackson
```

Exemplo com struct (I)

- O código a seguir demonstra o uso de uma estrutura:

```
#include <stdio.h>
#include <string.h>

struct curso {
  int id;
  char titulo[40];
  float carga_horaria;
};
```

Exemplo com struct (II)

```
int main() {
  struct curso c1 = {341279, "Introdução a C", 12.5};
  struct curso c2;
  /* inicializa c2 */
  c2.id = 341281;
  strcpy(c2.titulo, "C Avançado");
  c2.carga_horaria = 14.25;
  /* mostra as informações do curso */
  printf("%d\t%s\t%4.2f\n", c1.id, c1.titulo, c1.carga_ho
raria);
  printf("%d\t%s\t%4.2f\n", c2.id, c2.titulo, c2.carga_ho
raria);
  return 0;
}
```

Usando typedef

- A palavra-chave **typedef** cria uma definição de tipo que simplifica o código e facilita a leitura de um programa.
- **typedef** é comumente usado com registros struct porque elimina a necessidade de usar a palavra-chave **struct** ao declarar variáveis.

Exemplo com typedef

- Por exemplo:

```
typedef struct {
  int id;
  char titulo[40];
  float carga_horaria;
} curso;

curso c1;
curso c2;
```

Utilizando typedef

- Observe que uma marcação **struct** não é mais usada, em vez disso, um nome **typedef** aparece antes da declaração da estrutura.
- Agora, a palavra **struct** não é mais necessária nas declarações de variáveis, tornando o código mais limpo e fácil de ler.

Structs com structs

- Os membros de uma estrutura também podem ser estruturas.
- Por exemplo, considere as seguintes instruções:

```
typedef struct {
  int x;
  int y;
} ponto;

typedef struct {
  float radiano;
  ponto centro;
} circulo;
```

Structs com structs

- Chaves aninhadas são usadas para inicializar membros que são structs.
- O operador de ponto é usado duas vezes para acessar membros de membros, como nas instruções:

```
circulo c = {4.5, {1, 3}};
printf("%3.1f %d,%d", c.radiano, c.centro.x
, c.centro.y);
/* 4.5  1,3 */
```