MC-102 Algoritmos e Programação de Computadores
IC-UNICAMP

Aula 21 - Registros

Por: Luís Augusto Angelotti Meira
(Sala IC-71) 1S2005

Esta aula foi baseada em [1].

1 Objetivos

Conceituar tipos de dados definidos pelo usuário. Utilizar `struct` e `typedef` para criar registros.

2 Motivação

Dados que são relacionados podem ser salvos em uma única estrutura ou registro.

3 Aula e Exemplos

3.1 Definição

Estruturas são tipos de variáveis que agrupam dados geralmente desiguais; enquanto matrizes são tipos de variáveis que agrupam dados similares. Os itens de dados de uma estrutura são chamados **membros**, enquanto os itens de uma matriz são chamados **elementos**.

3.2 Como Declarar

Para se criar um registro, utiliza-se o comando `struct`. Sua forma geral é:

```
  struct nome_do_tipo_da_estrutura {
    tipo_1 nome_1;
    tipo_2 nome_2;
    tipo_3 nome_3;
    ...
    tipo_n nome_n;
} var_1, var_2;
```

Veja um exemplo de registro.

```
struct ficha {
  char nome[20];
  int idade;
  int RG;
} ficha1, ficha2;
```

3.3 Como Acessar

Para se acessar um campo de um registro, digita-se o nome da variável que contém o registro, por exemplo, `ficha1` seguida de um ponto, seguida do nome do capo.

Exemplo:

```
strcpy(ficha1.nome,"João da Silva";)
ficha1.idade = 5;
ficha1.RG = 567891011;
printf("Nome:%s\nidade %d\nRG %d\n",ficha1.nome,ficha1.idade, ficha1.RG);
```

Os comandos acima irão imprimir:

```
Nome:João da Silva
idade 5
RG 567891011
```

Para se acessar um elemento de um vetor:

```
ficha1.nome[3] = 'A';
```

3.4 Como Copiar

Retorna o valor inicial em `ficha1`:

```
strcpy(ficha1.nome,"João da Silva");
```

Errata: Diferentemente de Pascal, onde é necessário copiar campo a campo, em C é possível atribuir dois registros. Ex:

```
//***********************
ficha2 = ficha1;
//*********************
printf("Nome:%s\nidade %d\nRG %d\n",ficha2.nome,ficha2.idade, ficha2.RG);

}
```

cuja saída será:

```
Nome:João da Silva
idade 5
RG 567891011
```

Para se copiar um registro em outro existem mais duas possibilidades:

Copiar-se campo a campo ou copiar todos os campos em um único comando.

```
int tamanho;
strcpy(ficha2.nome,ficha1.nome);
ficha2.idade =   ficha1.idade;
ficha2.RG = ficha1.RG;
printf("Nome:%s\nidade %d\nRG %d\n\n\n",ficha2.nome,ficha2.idade, ficha2.RG);

// Limpa ficha2
strcpy(ficha2.nome,"NADA");
ficha2.idade =   0;
ficha2.RG = 0;

tamanho = sizeof(ficha1);
memcpy(&ficha2,&ficha1,tamanho);

printf("Nome:%s\nidade %d\nRG %d\n\n\n",ficha2.nome,ficha2.idade, ficha2.RG);
```

Temos que `sizeof(tipo )` é uma função que retorna o tamanho do `tipo` em bytes. É necessário, na função `memcpy` este tamanho, que é o números de bytes a ser copiado do segundo endereço para o primeiro.

A saída deste programa, considerando os dados inseridos, será:

```
Nome:João da Silva
idade 5
RG 567891011

Nome:João da Silva
idade 5
RG 567891011
```

A função `memcopy` copia os dados do endereço `&ficha1` para o endereço `&ficha2`. O número de bytes copiados é 28. 20 bytes do vetor nome, 4 bytes para idade e 4 bytes para RG.

4 Passando Estruturas para Funções

Estruturas podem ser passadas para funções da mesma forma que variáveis. Porém é necessário criar um tipo, através de `typedef`.

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

```
typedef struct venda {
  int pecas;
  float preco;
} venda ;

void insere( venda * A, int pecas, float preco){
  (*A).pecas = pecas;
  A-> preco = preco;
}

void imprime(venda C, venda D)
{
  printf("%d pecas %f reais\n", C.pecas, C.preco);
  printf("%d pecas %f reais\n", D.pecas, D.preco);
}

int main (){
  struct venda A, B;
  insere(&A,5,100);
  insere(&B,7,200);
  imprime (A,B);
  return 1;
}
```

Cuja saída será:

```
5 pecas 100.000000 reais
7 pecas 200.000000 reais
```

5 Funções Podem Retornar um Registro

Veja o exemplo:

```
#include <stdio.h>

typedef struct venda {
  int pecas;
  float preco;
} venda ;

venda novavenda( int pecas, float preco){
  venda A;
  A.pecas = pecas;
  A preco = preco;
  return A;
```

```
}

void imprime(venda C, venda D)
{
  printf("%d pecas %f reais\n", C.pecas, C.preco);
  printf("%d pecas %f reais\n", D.pecas, D.preco);
}

int main (){
  struct venda A, B;
  A = novavenda(5,100);
  B = novavenda(7,200);
  imprime (A,B);
  return 1;
}
```

5.1 Vetor de Registros e Registros Aninhados

É possível criar um vetor de registros. Exemplo:

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


struct tipo_endereco{
  int numero;
  int CEP;
  char logradouro[100];
  char bairro[100];
  char cidade[100];
  char estado[3];
};

int main(void){

  struct tipo_endereco endereco[10];

  endereco[0].numero = 100;
  endereco[0].CEP = 2345678;
  strcpy(endereco[0].logradouro,"Rua Jaçana");
  strcpy(endereco[0].bairro,"Perdizes");
  strcpy(endereco[0].cidade,"São Paulo");
   strcpy(endereco[0].estado,"SP");

  printf("%s n.o. %d \n%s %s %s\nCEP %d\n",endereco[0].logradouro,
   endereco[0].numero,
```

```
 endereco[0].bairro,
 endereco[0].cidade,
 endereco[0].estado ,
 endereco[0].CEP);

  memcpy(&endereco[1],&endereco[0],sizeof(struct tipo_endereco));

  endereco[1].numero = 200;
  endereco[1].CEP =  123123;

  printf("\n\n%s n.o. %d \n%s %s %s\nCEP %d\n",endereco[1].logradouro,
   endereco[1].numero,
 endereco[1].bairro,
 endereco[1].cidade,
 endereco[1].estado ,
 endereco[1].CEP);

}
```

Cuja saída será:

```
Rua Jaçana n.o. 100
Perdizes São Paulo SP
CEP 2345678


Rua Jaçana n.o. 200
Perdizes São Paulo SP
CEP 123123
```

É possivel criar registros aninhados. Exemplo:

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

struct tipo_endereco{
  int numero;
  int CEP;
  char logradouro[100];
  char bairro[100];
  char cidade[100];
  char estado[3];
};

struct tipo_ficha{
```

```
  char nome[100];
  struct tipo_endereco endereco;
};

struct tipo_repositorio{
  char nome_repositorio[100];
  int  tamanho;
  struct tipo_ficha ficha[100];
};

int main(void){
  struct tipo_repositorio repositorio;
  struct tipo_endereco endereco;
  repositorio.tamanho = 2;
  int i;
  strcpy(repositorio.nome_repositorio,"Catalogo Inicial");
  strcpy(repositorio.ficha[0].nome,"João");
  endereco.numero = 100;
  endereco.CEP = 2345678;
  strcpy(endereco.logradouro,"Rua Jaçana");
  strcpy(endereco.bairro,"Perdizes");
  strcpy(endereco.cidade,"São Paulo");
  strcpy(endereco.estado,"SP");
  memcpy(&repositorio.ficha[0].endereco,&endereco,sizeof(struct tipo_endereco));
  memcpy(&repositorio.ficha[1],&repositorio.ficha[0],sizeof(struct tipo_ficha));
  strcpy(repositorio.ficha[1].nome,"José");
  repositorio.ficha[1].endereco.numero = 200;
  repositorio.ficha[1].endereco.CEP = 123456;
  printf("%s\n",repositorio.nome_repositorio);
  for(i = 0 ; i < repositorio.tamanho; i++){
    printf("Ficha %d\n",i+1);
    printf("Nome %s\n",repositorio.ficha[i].nome);
    printf(" %s",repositorio.ficha[i].endereco.logradouro);
    printf("n.o. %d\n",repositorio.ficha[i].endereco.numero);
    printf(" %s",repositorio.ficha[i].endereco.bairro);
    printf(" %s",repositorio.ficha[i].endereco.cidade);
    printf(" %s\n",repositorio.ficha[i].endereco.estado);
     printf("CEP %d\n\n",repositorio.ficha[i].endereco.CEP);
  }

}
```

A saída do programa acima será:

```
Ficha 1
```

```
Nome João
 Rua Jaçanan.o. 100
 Perdizes São Paulo SP
CEP 2345678

Ficha 2
Nome José
 Rua Jaçanan.o. 200
 Perdizes São Paulo SP
CEP 123456
```

6 Tipos Enumerados

Pode-se definir tipos por meio da palavra chave `enum`, que ajuda a tornar mais clara a escrita do programa. O tipo da variável `enum` é sempre `int` e recebe automaticamente os valores $\{1, 2, 3, \ldots\}$.

veja o exemplo:

```
#include <stdio.h>
enum mes { Jan =1, Fev, Mar, Abr, Mai, Jun, Jul, Ago, Set, Out, Nov, Dez};
int main(){
  mes m1, m2;
  m1 = Abr;
  m2 = Jan;
  mes m3;
  m3 =(mes)(m2+m1);
  printf("%d %d %d\n",m1,m2,m3);

  if(m1 > m2 )
      printf("m1 %d \n",m1);
  else
      printf("m2 %d \n",m2);
  return 1;

}
```

Que terá como saída:

```
4 1 5
m1 4
```

O compilador associa automaticamente o valor do item i igual ao valor do item $i-1$ somado de um. O primeiro item recebe sempre zero. O programador pode atribuir um valor especifico a um dado item. No programa acima jan recebeu 1. O padrão é iniciar com zero. Os itens seguintes foram obtidos somando-se sempre 1. (Fev = 2, Mar = 3)

Veja o exemplo:

```
#include <stdio.h>
enum direcao {Leste=100, Norte=200, Oeste=350, Sul };
enum sexo {mas, fem};
enum chave {ON, OFF};
int main(){
  direcao  p1,p2,p3,p4,p5;
  p1 = Leste;
  p2 = Norte;
  p3 = Oeste;
  p4 = Sul;
  p5 = (direcao)(p1+p2);
  printf("%d %d %d %d %d\n",p1,p2,p3,p4,p5);
  chave key;
  key = ON;
  if(key == ON) {
    key = OFF;
    printf("desligado\n");
  }
  return 1;
}
```

Cuja saída será:

```
100 200 350 351 300
desligado
```

7 Exercícios

1) Assuma que as seguinte declarações tenham sido feitas:

```
struct corpo{
  float altura;
  float peso;
  };
struct corpo Joao;
```

a) Reescreva as instruções acima para definir a estrutura e declarar a variável de uma única vez.

b) Escreva uma instrução que indique que a altura de João é 1.68.

c) Escreva estruturas necessárias para definir o tipo "casal" contendo duas estruturas do tipo corpo.

d) escreva uma instrução necessária para escrever uma matriz com 10 estruturas do tipo casa;

e) Escreva as instruções necessárias para preencher o primeiro elemento da matriz anterior com os dados de Maria (alt=1.63, peso = 59.5) e de José (alt= 1.78, peso = 82.6).

3) Assuma que `st1, st2` e `st3` são variáveis de um mesmo tipo de estrutura. Quais instruções são validas:

a) st1 = st2;

b) st1 = st2 + st3;

c) st1 = st2 = st3;

d) st1 = st2 + 5;

5)

a) Escreva uma estrutura para conter três membros do tipo int chamados: hora, minuto e segundo. Atribua o nome tempo a esta estrutura.

b) Escreva uma estrutura para armazenar dados de um estacionamento. Ela deve ser capaz de armazenar o número da chapa do carro, a marca, a hora de entrada e a hora de saída do estacionamento. Utilize dois membros do tipo tempo, definido em "a)" para as horas de entrada e saída.

c) Crie uma variável com tipo igual a estrutura definida em "c)" e atribua valores a esta estrutura.

6)

a) Escreva uma instrução que defina um tipo enumerado `FRUTA` com nomes Pera, Maçã, Figo, Manga e Uva.

b) Declare variáveis do tipo `FRUTA` de nomes Ana e Joao e atribuas a ela os valores de Figo e Uva, respectivamente.

c) Quais atribuições são validas?

- Ana = Banana;
- Joal = Pera;
- Manga = Ana;
- Diferenca = Joao - Ana;

Referências

[1] Henrique José dos Santos. Curso de linguagem c, ufmg. Universidade Federal de Minas Gerais.