ESTRUTURA DE DADOS E ALGORITMOS I

Estrutura de Registros

Profº. Sérgio Roberto Costa Vieira, M.Sc.
Cursos de Computação
2º. Período

Definição

- Um registro é uma variável composta heterogênea.
  - É um conjunto de dados estruturados, os quais podem ser de tipos diferentes

- Os dados em um registro são representados através de variáveis ou constantes, normalmente chamadas de campos.
  - Para acessar esses campos usa-se o operador "." ponto

Vetor de Struct

- Para declarar um vetor de struct.
  - Primeiro definir a struct
  - Declarar o vetor do tipo struct criado

- Exemplo:

```
struct  aluno  turma[10];
```

Estrutura de Dados e Algoritmos I

Registros (Structs)

Exemplo

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

struct cad_aluno {
   char nome[40];
   float nota[4];
};

int main() {
   struct  cad_aluno  Aluno[3];
   int i, j;
   float Nota_Alu;

printf("\n\n Cadastro de Aluno \n\n");
for( j=0; j <= 2; j++ ){
   printf("Informe o nome do %do Aluno: ", j+1);
   fflush(stdin);
   gets(Aluno[ j ].nome);
for( i=0; i <= 3; i++ ) {
    printf("Informe a %da Nota: ", i+1);
    fflush(stdin);
    scanf(" %f ", &Nota_Alu);
     Aluno[ j ].nota[ i ] = Nota_Alu;
  }
}

for( j=0; j <= 2; j++ ){
 printf("\n Nome..: %s\n", Aluno[ j ].nome);
  for( i=0; i <= 3; i++ ) {
printf("%da Nota: %5.2f \n",i+1,Aluno[ j ].nota[ i ]);
   }
}
getch();
return 0;
}
```

Struct Aninhados

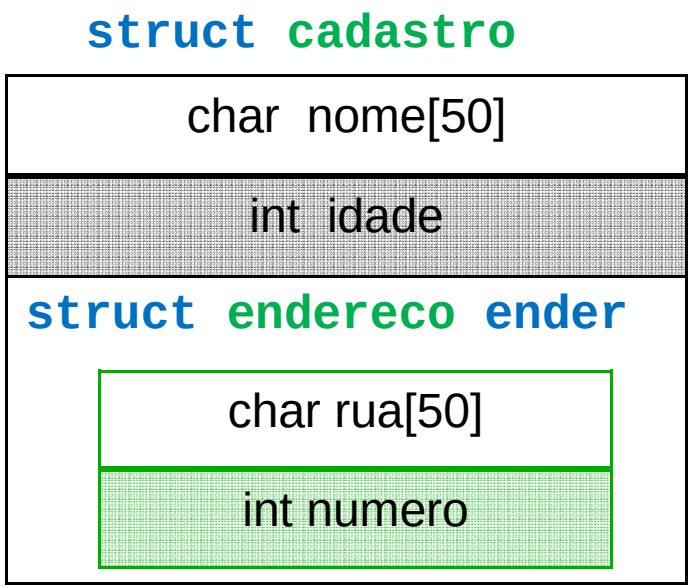
- Um registro pode agrupar diversas variáveis de tipos de dados diferentes.
  - É um tipo de dado com a diferença de que se trata de um tipo criado pelo programador.

- Sendo assim, pode-se declarar um registro que possua uma variável do tipo de outro registro:
  - denomina-se registros aninhados.

Struct Aninhados

```
struct endereco{
    char rua[50];
    int numero;
};

struct cadastro{
      char nome[50];
      int idade;
      struct endereco ender;
};
```



Estrutura de Dados e Algoritmos I

Registros (Structs)

Exemplo 1

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

struct endereco {
   char rua[40];
   int numero;
};

struct cadastro {
   char nome[50];
   int idade;
   struct endereco ender;
};

int main() {
   struct  cadastro  C;

printf("\n\n LEITURA DOS DADOS: ");
printf("\n\n Informe o nome: ");
gets(C.nome);
printf("\n\n Informe a idade: ");
scanf ("%d ",&C.idade);

printf("\n\n Informe a rua onde mora: ");
fflush(stdin);
gets (C.ender.rua);

printf("\n\n Informe o numero da casa: ");
scanf ("%d ",&C.ender.numero);


printf("\n\n IMPRESSAO DOS DADOS: ");
printf("\n Seu nome: %s", C.nome );
printf("\n Sua idade: %d", C.idade );
printf("\n Rua: %s", C.ender.rua );
printf("\n Numero: %d", C.ender.numero );


getch();
return 0;
}
```

Estrutura de Dados e Algoritmos I

Registros (Structs)

Exemplo 2

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

struct tipo_endereco {
   char  rua[50];
   int  numero;
   char  bairro[20];
   char  cidade[30];
   char  estado[3];
   int  CEP;
};

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

struct ficha_pessoal {
   char  nome[50];
   int  telefone;
   struct tipo_endereco endereco;
   struct nascimento dtnasc;
} ficha[3] ;

int  main(){
   int i=0;
   for( i=0; i < 3; i++ ){
        printf("\nNome");
        gets(ficha[ i ].nome);
        printf("\nRua");
        gets(ficha[ i ].endereco.rua);
```

Exemplo 3 - *continuação*

```
  printf("\nDia de Nascimento");
  scanf("%d", &ficha[ i ].dtnasc.dia);
  printf("\nMes de Nascimento");
  scanf("%d", &ficha[ i ].dtnasc.mes);
  printf("\nAno de Nascimento");
  scanf("%d", &ficha[ i ].dtnasc.ano);
  fflush(stdin);

  strcpy(ficha[ i ].endereco.estado,"PA");
}
printf("\n\n Ficha Pessoal \n");

for( i=0; i < 3; i++ ){
  printf("\nNome: %s", ficha[ i ].nome);
  printf("\nRua: %s", ficha[ i ].endereco.rua);
  printf("\nUF: %s", ficha[ i ].endereco.estado);
  printf("\nNascimento: %d / %d / %d: " ,
ficha[ i ].dtnasc.dia, ficha[ i ].dtnasc.mes,
ficha[ i ].dtnasc.ano);
}

printf("\n\n");
getch();
return 0;
}
```

Comando TYPEDEF

- A linguagem C permite que o programador defina os seus próprios tipos com base em outros tipos de dados existentes.

- Para isso, utiliza-se o comando typedef:

```
typedef  tipo_existente  novo_nome.
```

Comando TYPEDEF

- O comando typedef **NÃO** cria um novo tipo.

- Ele apenas permite que você defina um sinônimo para um tipo já existente:

```
typedef  int  inteiro;
```

- Foi criado um sinônimo para o tipo de dados **int**.

Exemplo

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

typedef  int  inteiro;

int main() {
   int  x = 10;
   inteiro  y = 20;

   y = y + x;

   printf("\nSoma: = %d", y);

getch();
return 0;
}
```

Comando TYPEDEF

- O comando typedef pode ser combinado com a declaração de um tipo definido pelo programador (struct, union etc) em uma única instrução.

```
typedef struct cadastro{
    char nome[50];
    int idade;
    char rua[50];
    int numero;
} Dados;

  Dados cad1, cad2;
```

Estrutura de Dados e Algoritmos I

Registros (Structs)

Exemplo

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

typedef  struct CadastroAluno {
   char  nome[50];
   int  codigo, idade;
} Aluno;

int main() {
 Aluno A[ 3 ];

for( int  i=0; i < 3; i++ ) {
 printf("Informe os dados do %do Aluno:", i+1);
 printf("\n Digite o nome: ");
 fflush(stdin);
 gets(A[ i ].nome);
 printf("\n Digite o codigo: ");
 scanf("%d", &A[ i ].codigo);
 printf("\n Digite a idade: ");
 scanf("%d", &A[ i ].idade);
}

for( int  i=0; i < 3; i++ ) {
    printf("\n\n%do  Aluno: ", i+1);
    printf("Codigo: %d  Nome: %s
Idade: %d ", A[ i ].codigo,  A[ i ].nome,
A[ i ].idade );
}

getch();
return 0;
}
```

Comando TYPEDEF

- As três formas possíveis para declarar um novo tipo Data com ou sem typedef são:

```
struct Data{
   int Dia, Ano;
   char mes[50];
};

int main(){
   struct Data d1;
   d1.Dia = 26;
   d1.Mes = “Jan”;
   d1.Ano = 1993;
}
```

```
struct Data{
   int Dia, Ano;
   char mes[50];
};

typedef struct Data DT;
int main(){
   DT d1;
   d1.Dia = 26;
   d1.Mes = “Jan”;
   d1.Ano = 1993;
}
```

```
typedef struct Data{
   int Dia, Ano;
   char mes[50];
}DT;

int main(){
   DT d1;
   d1.Dia = 26;
   d1.Mes = “Jan”;
   d1.Ano = 1993;
}
```

Estrutura de Dados e Algoritmos I

Registros (Structs)

Exemplo

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

typedef  struct Dados {
   int RA;
   double nota;
} Aluno;

int main() {
 Aluno turma[ 5 ];
 double media;

for( int  i=0; i < 5; i++ ) {
 printf("Informe dados do %do Aluno:", i+1);
 printf("\n Digite o RA do aluno: ");
 scanf("%d", &turma[ i ].RA);
 printf("\n Digite a media do aluno: ");
 scanf("%f", &turma[ i ].nota);
}

media = 0.0;
for( int  i=0; i < 5; i++ ) {
    media = media + turma[ i ].nota;
}

media = media/5.0;

printf("\nA media da turma e: %f\n", media);

getch();
return 0;
}
```

Passagem de Structs como Parâmetros

- As structs contém dentro de si outras variáveis.
- Sendo assim, ela pode ser passada para uma função de duas formas distintas:
  - Toda a estrutura;
  - Apenas determinados campos da estrutura.
- As regras de passagem de parâmetros são as mesmas para structs, enum e union.

Estrutura de Dados e Algoritmos I

Registros (Structs)

Passagem de uma Estrutura como Parâmetro por Valor

Exemplo

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

struct dado {
   int  x, y;
};

void  imprime(struct dado p) {
   printf(" x = %d \n", p.x);
   printf(" y = %d \n", p.y);
}
```

```
int main() {

   struct  dado  p1 = {10, 20};

    imprime( p1 );

getch();
return 0;
}
```

Estrutura de Dados e Algoritmos I

Registros (Structs)

Passagem de um Campo como Parâmetro por Valor

Exemplo

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

struct dado {
   int  x, y;
};

void  imprime(int  n) {
   printf(" Valor = %d \n", n);

}
```

```
int main() {

   struct  dado  p1 = {10, 20};

    imprime( p1.x );

    imprime( p1.y );

getch();
return 0;
}
```

Passagem de Structs como Parâmetros

- Passagem de Parâmetros por Referência
- Atenção a alguns cuidados que devem ser tomados ao acessar os campos dentro da função, siga estes passos:
  - Utilizar o "*" na frente do nome da variável da estrutura;
  - Colocar o "*" e o nome da estrutura entre parênteses ().
  - Por fim, acessar o campo da estrutura utilizando o operador ponto ".".

Estrutura de Dados e Algoritmos I

Registros (Structs)

Passagem de uma Estrutura como Parâmetro por Referência

Exemplo

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

struct dado {
   int  x, y;
};

void  atribui(struct dado *p) {
   (*p).x = 10;
   (*p).y = 20;
}
```

```
int main() {

   struct  dado  p1;

    atribui( &p1 );

   printf(" x = %d \n", p1.x);
   printf(" y = %d \n", p1.y);


getch();
return 0;
}
```

Passagem de Structs como Parâmetros

- Passagem de Parâmetros por Referência
- O operador seta(->) substitui o uso do conjunto dos operadores “*” e “.” no acesso ao campo de uma estrutura.
- O operador seta(->) é utilizado quando uma referência para uma estrutura(struct) é passada para uma função.

Estrutura de Dados e Algoritmos I

Registros (Structs)

Passagem de uma Estrutura como Parâmetro por Referência com Operador seta

Exemplo

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

struct dado {
   int  x, y;
};

void  atribui(struct dado *p) {
   p -> x = 10;
   p -> y = 20;
}
```

```
int main() {

   struct  dado  p1;

    atribui( &p1 );

   printf(" x = %d \n", p1.x);
   printf(" y = %d \n", p1.y);


getch();
return 0;
}
```

Estrutura de Dados e Algoritmos I

Registros (Structs)

Passagem de um Campo como Parâmetro por Referência

Exemplo

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

struct dado {
   int  x, y;
};

void  soma_imprime(int  *n) {
   *n = *n + 1;
    printf(" Valor = %d \n", *n);

}
```

```
int main() {

   struct  dado  p1 = {10, 20};

    soma_imprime( &p1.x );

    soma_imprime( &p1.y );

getch();
return 0;
}
```

Estrutura de Dados e Algoritmos I

Registros (Structs)

Passagem de uma Estrutura como Parâmetro por Referência

Exemplo com Arrays

```
#include<stdio.h>
#include<conio.h>
#define TAM 3

typedef struct CadastroAluno{
   char nome[30];
   char turma[3];
   char nota1[3];
} aluno;

void impressao(aluno *a){
   int i;
   for(i=0; i<TAM; i++){
     printf("\n NOME: %s", a[i].nome);
     printf("\n Nota%d: %s", i+1, a[i].nota1);
   }
}
```

```
void cadastroAluno(aluno *a){
   int i;
   for(i=0; i<TAM; i++){
     printf("NOME: ");
     gets(a[i].nome);
     printf("Nota 1: ");
     gets(a[i].nota1);
   }
}

int main(){
    aluno alunos[TAM], *p;
    p = &alunos[0];

    cadastroAluno(p);
    impressao(alunos);

 getch();
 return 0;
}
```

Estrutura de Dados e Algoritmos I

Registros (Structs)

Retornando uma Estrutura por meio de uma Função

Exemplo de Estrutura por Retorno

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

typedef struct DadosVenda{
        int peca;
        float preco;
}Venda;

Venda TotalDeVendas(Venda X, Venda Y){
   Venda T;
    T.peca = X.peca + Y.peca;
   T.preco = X.peca*X.preco + Y.peca*Y.preco;
    return T;
}
```

```
int main(){
  Venda A, B, Total;

  printf("Venda A======\n");
  printf("Digite o numero de pecas: ");
  scanf("%d", &A.peca);
  printf("\nDigite o preco: ");
  scanf("%f", &A.preco);

  printf("\nVenda B======\n");
  printf("Digite o numero de pecas: ");
  scanf("%d", &B.peca);
  printf("\nDigite o preco: ");
  scanf("%f", &B.preco);

  Total = TotalDeVendas(A, B);

  printf("\nTotal das Vendas======\n");
  printf("Total de pecas: %d ", Total.peca);
  printf("\nPreco Total : %.2f ", Total.preco);

 return 0;
}
```

Exercício 1

Fazer um programa que cria uma estrutura livro, que contém os elementos título, ano de edição, número de páginas e preço. Criar uma variável desta estrutura que é um vetor de 5 elementos. Ler os valores para a estrutura e imprimir a média do número de páginas do livros:

Exercício 2

Fazer um programa que considere uma estrutura para armazenar algumas informações dos pacientes de uma clínica médica (nome, data de nascimento, identidade, sexo, endereço). Os campos data de nascimento e endereço devem ser uma nova estrutura separada da estrutura principal pacientes. Criar uma opção para imprimir os dados dos pacientes por sexo, informando nome e idade dos pacientes. Após criar outra opção para imprimir todos os dados dos pacientes por classificação alfabética:

ESTRUTURA DE DADOS E ALGORITMOS I

Estrutura de Registros

Profº. Sérgio Roberto Costa Vieira, M.Sc.
Cursos de Computação
2º. Período