ESTRUTURAS

Profa. Fabrícia Damando Santos
fabriciadamando@gmail.com

Introdução: Tipo estrutura

- Algumas vezes não conseguimos manusear dados em variáveis escalares como os inteiros e ponto flutuante.
- Estes tipo de dados são compostos por vários dos tipos básicos dos C.
- As estruturas permitem uma organização dos dados dividida em campos e registros.
- É um tipo de dado com campos compostos de tipos mais simples
- Os elementos são acessados através do operador de acesso "ponto" (.)

STRUCT - EXEMPLO

```
struct ponto /* declara ponto do tipo struct */
{
    float x;
    float y;
};
...
struct ponto p; /* declara p como variável do tipo struct ponto */
...
p.x = 10.0; /* acessa os elementos de ponto */
p.y = 5.0;
```

SINTAXE

```
struct lapis {
  int dureza;
  char fabricante;
  int numero;
};
main (){
 int i;
 struct lapis p[3];
 p[0].dureza=2;
 p[0].fabricante='f';
 p[0].numero=482;

 p[1].dureza=3;
 p[1].fabricante='g';
 p[1].numero=483;

 p[2].dureza=21;
 p[2].fabricante='x';
 p[2].numero=15;

printf("Dureza          Fabricante                Numero\n");
for(i=0; i<3;i++){
printf("%d\t%c\t\t%d\n",p[i].dureza,p[i].fabricante,p[i].numero);
}
}
```

Struct que captura as coordenadas de um ponto qualquer

```
/* Captura e imprime as coordenadas de um ponto qualquer */
#include <stdio.h>
struct ponto
{
    float x;
    float y;
};

int main (void)
{
    struct ponto p;
    printf("Digite as coordenadas do ponto(x y): ");
    scanf("%f %f", &p.x, &p.y);
    printf("O ponto fornecido foi: (%.2f,%.2f)\n", p.x, p.y);
    return 0;
}
```

```
struct ALUNO
{
char nome [40];
int registro ;
int ano_entrada ;
char curso [20];
};
```

- O exemplo acima não alocou espaço de memória já que nenhuma variável foi realmente definida.
- Esta declaração é apenas um modelo de como estruturas do tipo ALUNO devem ser construídas.
- Para definir estruturas deste tipo podemos usar a seguinte declaração:
  - struct ALUNO paulo, carlos, ana;

- Nesta declaração três estruturas do tipo ALUNO foram criadas.
- ***Esta declaração alocou espaço para armazenar os dados dos três alunos***.

- Como ocorre com as variáveis, as estruturas também podem ser referenciadas por ponteiros.

- Assim, definindo-se por exemplo o ponteiro *p para a estrutura apresentada (lapis), pode-se usar a sintaxe (*p).dureza.

- Porém, para referenciar o ponteiro há ainda outra sintaxe, através do operador -> , como por exemplo, p->dureza.

Ponteiros para estruturas

- Ponteiros para estruturas:
- Acesso ao valor de um campo x de uma variável estrutura p: p.x
- Acesso ao valor de um campo x de uma variável ponteiro pp: pp->x

```
struct ponto *pp;

/* formas equivalentes de acessar o valor de um campo x */

(*pp).x = 12.0;
pp->x = 12.0;
```

STRUCT – COM FUNÇÃO

```
/* função que imprima as coordenadas do ponto */

void imprime (struct ponto p)
{
printf("O ponto fornecido foi: (%.2f,%.2f)\n", p.x, p.y);
}
```

STRUCT – FUNÇÃO – PASSAGEM DE PARÂMETROS

- Passagem de estruturas para funções:
  - Referência:
    - apenas o ponteiro da estrutura é passado, mesmo que não seja necessário alterar os valores dos campos dentro da função
    - Ex:

```
/* função que imprima as coordenadas do ponto */
void imprime (struct ponto* pp)
{
   printf("O ponto fornecido foi: (%.2f,%.2f)\n", pp->x, pp->y); }

void captura (struct ponto* pp) {
printf("Digite as coordenadas do ponto(x y): ");
scanf("%f %f", &p->x, &p->y); }
int main (void) {
struct ponto p;
captura(&p);
imprime(&p);
return 0;  }
```

STRUCT -ALOCAÇÃO DINÂMICA

- Alocação dinâmica de estruturas:
  - tamanho do espaço de memória alocado dinamicamente é dado pelo operador **sizeof** aplicado sobre o tipo estrutura
  - função **malloc** retorna o endereço do espaço alocado, que é então convertido para o tipo ponteiro da estrutura
  - Ex:

```
struct ponto* p;

p = (struct ponto*) malloc (sizeof(struct ponto));
...
p->x = 12.0;
...
```

Definição de novos tipos

- Typedef
  - permite criar nomes de tipos
  - útil para tratar tipos complexos

- Ex:

```
// estrutura ponto
typedef struct
 {
    float x;
    float y;
 } ponto;

// ponteiro para estrutura ponto
ponto *ptponto;
```

Exemplo – construção de tipos struct

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

// Construção de Tipos

typedef struct
{
   char nome[25];
   int idade;
} empregado;


int main()
{
   empregado funcionario;

   printf("Nome:");
   scanf("%s“,funcionario.nome);
   printf("Idade:");
   scanf("%d",&funcionario.idade);

   system("PAUSE");
   return 0;
}
```

Exemplo

```
#include < stdio .h>

typedef struct _EMPREGADO
{
char nome [40];
float salario ;
} EMPREGADO ;

int main ()
{
EMPREGADO temp , emp1 ;
puts (" Entre com nome .");
gets ( emp1 . nome );
puts (" Qual o salario ?");
scanf("%f", & emp1 . salario );
temp = emp1 ;
printf ("O salario de %s e %.2 f\n" ,
temp .nome , temp . salario );
return 0;
}
```

Tipo enumeração

- Enum
  - declara uma enumeração, ou seja, um conjunto de constantes inteiras com nomes que especifica os valores legais que uma variável daquele tipo pode ter
  - oferece uma forma mais elegante de organizar valores constantes
  - Ex:

```
typedef enum
{
    TRUE = 1,
    FALSE = 0
} bool;

bool  resultado;
```