Computação para Informática - Prof. Adriano Joaquim de Oliveira Cruz
Aula Prática sobre Estruturas

O objetivo desta aula prática é exercitar estruturas

1 Introdução

Uma estrutura é um conjunto de uma ou mais variáveis, que podem ser de tipos diferentes, agrupadas sob um único nome. As variáveis que compõem a estrutura são os seus membros, elementos ou campos. A melhor maneira de declarar estruturas é usar **typedef**. Por exemplo, o comando a seguir define um tipo chamado `ALUNO`.

```
typedef struct _ALUNO  {
     char nome[TAMNOME];
     float salario;
     float imposto;
} ALUNO ;
```

A definição de variáveis do tipo aluno agora pode ser feita do modo.

```
ALUNO paulo, carlos, ana, turma[100];
```

Para referenciar um elemento da estrutura usa-se o nome da variável do tipo da estrutura seguida de um ponto e do nome do elemento. Por exemplo, `paulo.ano_entrada = 1999;` armazena o ano em que aluno `paulo` entrou na universidade. Para ler o ano em que `ana` entrou na Universidade usa-se o seguinte comando: `scanf("%d", &ana.ano_entrada);` Para imprimir o salário de `carlos` usa-se o seguinte comando: `printf("%f\n", carlos.salario);`

Uma estrutura pode conter vários tipos de variáveis inclusive outras estruturas. Por exemplo, considere as seguintes estruturas:

```
typedef struct _PONTO {
    float x, y;
} PONTO;

typedef struct _CIRCUNFERENCIA {
    float raio;
    PONTO centro;
} CIRCUNFERENCIA;
```

Estas estruturas definem elementos geométricos em um espaço de duas dimensões. Observe que a estrutura `CIRCUNFERENCIA` tem como membro uma variável do tipo `PONTO`.

O programa 1 mostra como ler as coordenadas de um ponto.

O programa 2 mostra como ler as coordenadas do centro de uma circunferência.

Listing 1: Exemplo de estrutura.

```
#include<stdio.h>
typedef struct _PONTO {
    float x, y;
} PONTO;
int main (void) {
     PONTO p;

     printf("Entre com as coordenadas do ponto p\n");
     scanf("%f %f", &p.x, &p.y);
     printf("Dados lidos\n");
     printf("Ponto p: x = %f, y = %f\n", p.x, p.y);
     return 0;
}
```

2 Exercícios

Exercício 1: Escreva o programa mostrado no exemplo 1.

Exercício 2: Escreva o programa mostrado no exemplo 2.

Exercício 3: Escreva um programa que leia as coordenadas de dois pontos `p1` e imprima a distância entre eles.

Exercício 4: Escreva um programa que leia as coordenadas de um ponto `p1` e os dados de uma circunferência `c1`. Imprima se o ponto está contido dentro da circunferência.

Exercício 5: Considere a seguinte estrutura:

```
typedef struct _TEMPO {
    int hora, minuto, segundo;
} TEMPO;
```

Escreva um programa que leia dois tempos (`TEMPO t1, t2;`) gastos em uma tarefa qualquer. O programa deve imprimir o maior tempo.

Exercício 6: Considere a seguinte estrutura:

```
typedef struct _TEMPO {
    int hora, minuto, segundo;
} TEMPO;
```

Escreva um programa que leia dois tempos (`TEMPO t1, t2;`) gastos em uma tarefa qualquer. O programa deve somar estes tempos armazenar o resultado em um terceiro (`TEMPO t3`). Imprima o resultado da soma.

Exercício 7: Escreva uma função `comparaTempo(TEMPO t1, TEMPO t2);` que compara dois tempos `t1` e `t2` gastos na execução de tarefas. A função retorna um valor de acordo com as seguintes regras:

$$\begin{cases} valor < 0 & \text{se } \texttt{t1 < t2} \\ valor = 0 & \text{se } \texttt{t1 = t2} \\ valor > 0 & \text{se } \texttt{t1 > t2} \end{cases} \quad (1)$$

Listing 2: Exemplo de estrurura dentro de estrutura.

```
#include<stdio.h>
typedef struct _PONTO {
    float x, y;
} PONTO;
typedef struct _CIRCUNFERENCIA {
    float raio;
    PONTO centro;
} CIRCUNFERENCIA;
int main (void) {
     CIRCUNFERENCIA c1;

     printf("Entre com o raio do circulo c1\n");
     scanf("%f", &c1.raio);
     printf("Entre com as coordenadas do centro do circulo c1\n");
     scanf("%f %f", &c1.centro.x, &c1.centro.y);
     printf("Dados lidos\n");
     printf("Circulo c1: raio = %f, x = %f, y = %f\n", c1.raio, c1.centro.x,
                         c1.centro.y);
     return 0;
}
```

Modifique o programa do exercício que imprime o maior tempo para que ele use esta função.

Exercício 8: Escreva um programa que leia os tempos gastos em 50 tarefas e os imprima em ordem crescente de tempo gasto.

Exercício 9: (Desafio) Escreva um programa que leia os dados de dois circunferências e verifique se elas estão:

- uma interna a outra;
- tangentes externamente;
- secantes;
- externas uma a outra.

Exercício 10: Considere a seguinte estrutura:

```
typedef struct _JOGADOR {
    int pontos;
     char nome[42];
} JOGADOR;
```

Escreva um programa que crie um vetor com os dados de 5 jogadores; leia estes do teclado e os imprima na ordem em que foram lidos.

Exercício 11: Modifique o programa anterior de modo que os dados dos jogadores sejam impressos em ordem decrescente dos valores dos pontos.

Exercício 12: Modifique o programa anterior de modo que após a impressão dos dados dos jogadores seja lido o dado de mais um recordista e o jogador com o menor número de pontos seja retirado da lista. A nova lista deve ser impressa.

Exercício 13: Considere a seguinte estrutura:

```
typedef struct _FRACAO {
    int numerador, denominador;
} FRACAO;
```

Escreva um programa que leia duas frações e calcule e imprima sua:

1. soma;
2. subtração;
3. produto;
4. divisão

Caso uma fração com denominador igual a zero seja lido o programa deve emitir um aviso e parar.

Exercício 14: Modifique o programa anterior para que as operações sejam realizadas pelas seguintes funções:

```
FRACAO soma (FRACAO a, FRACAO b); /* a + b */
FRACAO subtracao (FRACAO a, FRACAO b); /* a - b */
FRACAO multiplicacao (FRACAO a, FRACAO b); /* a * b */
FRACAO divisao (FRACAO a, FRACAO b); /* a / b */
```
