Dados estruturados, Ponteiros e Uso de Função com Ponteiro

Tipos estruturados

Em C, podemos definir um tipo de dado cujos campos são compostos de vários valores de tipos mais simples.
Para ilustrar, vamos considerar o desenvolvimento de programas que manipulam pontos no plano cartesiano.
Cada ponto pode ser representado por suas coordenadas x e y, ambas dadas por valores reais. Sem um mecanismo para agrupar as duas componentes, teríamos que representar cada ponto por duas variáveis independentes.

```
float x;
float y;
```

No entanto, deste modo, os dois valores ficam dissociados e, no caso do programa manipular vários pontos, cabe ao programador não misturar a coordenada x de um ponto com a coordenada y de outro.
Para facilitar este trabalho, a linguagem C oferece recursos para agruparmos dados.
Uma estrutura, em C, serve basicamente para agrupar diversas variáveis dentro de um único contexto.

No nosso exemplo, podemos definir uma estrutura ponto que contenha as duas variáveis. A sintaxe para a definição de uma estrutura é mostrada abaixo:

```
struct ponto {
   float x;
   float y;
}
```

Desta forma, a estrutura ponto passa a ser um tipo e podemos então declarar variáveis deste tipo.

struct ponto p;

Esta linha de código declara p como sendo uma variável do tipo struct ponto.
Os elementos de uma estrutura podem ser acessados através do operador de acesso "ponto" (.). Assim, é válido escrever:

```
ponto.x = 10.0;
ponto.y = 5.0;
```

Manipulamos os elementos de uma estrutura da mesma forma que variáveis simples.
Podemos acessar seus valores, atribuir-lhes novos valores, acessar seus endereços, etc.

```
#include <stdio.h>

struct ponto {
	float x;
	float y;
};

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

Exercícios:

1. Construa um programa que define uma estrutura de tipo pessoa.

2. Construa um programa que solicite inserção de dados para a estrutura de tipo pessoa.

3. Construa um programa que imprime a estrutura tipo pessoa.

Definição de Novos Tipos (typedef)

Este tipo de estrutura permite criar nomes de tipos e útil para abreviar nomes de tipos e para tratar tipos complexos.

```
typedef unsigned char UChar
typedef float Vetor[4];

Vetor v; /*exemplo de declaração usando Vetor*/

V[0] =3;
```

UChar o tipo char sem sinal
Vetor um tipo que representa um vetor de quatro elementos

Exemplo: definição de nomes de tipos para as estruturas

```
struct ponto{
```

```
  float x;
  float y;
};

typedef struct ponto Ponto;
```

ponto representa uma estrutura com 2 campos do tipo float
Ponto representa a estrutura ponto

Exemplo: (definição em um comando só)

```
typedef struct ponto{
  float x;
  float y;
} Ponto;
```

Exercícios:

4. Construa um programa que define um tipo novo de dados, para uma estrutura de um animal, deverá conter atributos como: raça, cor, peso e sexo. E solicite ao usuário do programa 10 espécies e depois imprima em tela.

5. Construa um programa que define uma estrutura de peça, que poderá ser usado para outros programas, deverá conter atributos como: código, nome, cor, peso, preço. Também uma função entrada de dados e uma função de saídas de dados.

Uso de ponteiro em C ANSI

Ponteiro é basicamente um apontador para algo, uma analogia sobre ponteiro são os endereços de ruas de uma cidade. Imagine que você quer ir à casa de Pedro, só ira conseguir se você tiver qual é a cidade que ele mora, o bairro, o endereço e o número, pois estas são as coordenadas essenciais para chegar a casa de Pedro.
Já em C ANSI, ponteiro é o nome da variável que indica o que está armazenado nela, ou seja, representa o endereço de memória onde informa as coordenadas da variável (o endereço de uma variável é um ponteiro).
Em C ANSI quando queremos representar um ponteiro usamos o * no início do nome da variável.

Estrutura:

[tipo de variável] * [nome da variável];

Os ponteiros são usados para diversas atividades avançadas, como:

1. Fornecerem maneiras com as quis as funções podem realmente modificar os argumentos que recebem (passagem por referencia);
2. Passarem dados do tipo de matrizes e strings mais convenientemente de uma função para outra (usá-los no lugar de matrizes);
3. Manipular os elementos de matrizes mais facilmente, por meio de movimentação de ponteiros para eles (ou parte delas), no lugar de índices entre colchetes;
4. Criar estrutura de dados complexa, como listas encadeadas e arvores binárias, em que um item deve conter referências a outros;
5. Alocar e deslocar memória dinamicamente do sistema;
6. Passar para uma função o endereço de outra função;

Exemplo:

```
#include <stdio.h>

int main (void){
   int a = 3, *pa;
   printf("&a = %p \t  a = %i\n",&a, a);
   printf("pa = %p \t pa = %i\n\n",pa, *pa);
   pa = &a;
   printf("&a = %p \t  a = %i\n",&a, a);
   printf("pa = %p \t pa = %i\n\n",pa, *pa);
   *pa = 6;
   printf("&a = %p \t  a = %i\n",&a, a);
   printf("pa = %p \t pa = %i\n\n",pa, *pa);
   a = 5;
   printf("&a = %p \t  a = %i\n",&a, a);
   printf("pa = %p \t pa = %i\n\n",pa, *pa);
   getch();
}
```

Exercícios:

6. Construa um programa para calcular uma equação de segundo grau, onde x' e x'' deverão ser um ponteiro, imprima o valor de x'e x'' após a execução do programa.

7. Construa um programa para calcular a área e de um circulo, a área deve ser do tipo de ponteiro.

Passando ponteiros para funções

Os ponteiros oferecem meios de alterarmos valores de variáveis acessando-as indiretamente. Sabemos que as funções não podem alterar diretamente valores de variáveis da função que fez a chamada. No entanto, se passarmos para uma função os valores dos endereços de memória onde suas variáveis estão armazenadas, a função pode alterar, indiretamente, os valores das variáveis da função que a chamou.

Exemplo:

```
#include <stdio.h>
void troca (int *px, int *py){
    int temp;
    temp = *px;
    *px = *py;
    *py = temp;
}

int main (void){
   int a = 5, b = 7;
   troca(&a, &b); /* passamos os endereços das variáveis */
   printf("a:%d b:%d \n", a, b);
   getch();
   return 0;
}
```

Passagem de vetores para funções

Passar um vetor para uma função consiste em passar o endereço da primeira posição do vetor. Se passarmos um valor de endereço, a função chamada deve ter um parâmetro do tipo ponteiro para armazenar este valor. Assim, se passarmos para uma função um vetor de int, devemos ter um parâmetro do tipo int*, capaz de armazenar endereços de inteiros. Salientamos que a expressão "passar um vetor para uma função" deve ser interpretada como "passar o endereço inicial do vetor". Os elementos do vetor não são copiados para a função, o argumento copiado é apenas o endereço do primeiro elemento.

Exemplo:

```
#include <stdio.h>
/* Função para cálculo da média */
float media (int n, float* v){
   int i;
   float s = 0.0;
   for (i = 0; i < n; i++)
      s += v[i];
```

```
   return s/n;
}

int main (void){
   float v[10];
   float med;
   int i;
   /* leitura dos valores */
   for ( i = 0; i < 10; i++ ){
      printf("Digite o valor %i: ",i+1);
      scanf("%f", &v[i]);
   }
   med = media(10,v);
   printf ( "\nMedia = %.2f ", med);
   getch();
   return 0;
}
```

Exercícios:

8. Construa um programa que contém duas variáveis inteiras, e cada uma destas variáveis deverá ter um valor atribuído nelas, através de uma entrada de dado do usuário. Também elaborem no mesmo programa três funções, que deveram usar o conceito de ponteiro para manipular a passagem dos dados para as funções. As funções deverão ter as seguintes funcionalidades: a soma das duas variáveis, a diferença entre as duas variáveis e a multiplicação das duas variáveis.

12. Faça um programa que peça um número para calcular o fatorial. Deve tem uma função para calcular o fatorial (por referência).

13. Faça um programa para colocar três números em ordem crescente. Deve ter uma função para fazer a troca dos valores, use uma função usando passagem de parâmetro por referência.

14. Faça uma função que receba como parâmetros por referência a linha, a coluna e uma mensagem, esta função deve imprimir na tela a mensagem na posição solicitada.

15. Faça uma função que leia um nome, converta o primeiro caractere deste nome para maiúscula. O nome deve ser passado para função por referência.

16. Faça uma função que leia um texto e converta este texto para maiúscula. O texto deve ser passado para função por referência.

17. Faça uma função que calcule a média de uma sequência de valores, os valores deverão ser passados por meio de um vetor de referência.