Estrutura de Dados

Exemplo: Manipula Números Pares

Profa. Gina M. B. Oliveira

Exercício 1

Construa um programa que manipula um vetor de inteiros (suponha tamanho do vetor igual a 5). Os números iniciais do vetor devem ser digitados pelo usuário. Posteriormente, o programa manipula esses dados de forma que todo número par deve ser dividido por 2. Ao final, imprimir o vetor resultante.

Solução 1: Programa monolítico (sem uso de funções)

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

int main()
{
    int N=5,i,temp;
    int numero[5];

   for (i=0;i<N;i++) {
        printf("\nDigite o %do. numero:", i+1);
        scanf("%d",&numero[i]);
    }

    for (i=0;i<N;i++) {
        if ((temp=numero[i] / 2) == numero[i]/2.0) {
                numero[i]=temp;
        }
    }

    for (i=0;i<N;i++) {
        printf("\nO %do. numero do vetor resultante eh: %d",i+1,numero[i]);
    }

    return 0;
}
```

Exercício 2

Construa um programa que manipula um vetor de inteiros (suponha tamanho do vetor igual a 5). Os números iniciais do vetor devem ser digitados pelo usuário. Posteriormente, o programa manipula esses dados de forma que todo número par deve ser dividido por 2. Ao final, imprimir o vetor resultante.

Solução 2: Uso de uma função manipula_pares, que recebe o vetor e altera os elementos pares.

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

void manipula_pares(int *vetor, int N);

int main()
{
   int N=5, i;
   int numeros[5];

   for (i=0;i<N;i++) {
      printf("\nDigite o %do. numero:", i+1);
      scanf("%d",&numeros[i]);
   }
   manipula_pares(numeros,N);
   for (i=0;i<N;i++) {
      printf("\nO %do. numero do vetor resultante eh: %d",i+1,numeros[i]);
   }

   return 0;
}

void manipula_pares(int *vetor, int N) {
int i, temp;

   for (i=0;i<N;i++) {
      if ((temp=vetor[i] / 2) == vetor[i]/2.0) {
         vetor[i]=temp;
      }
   }
}
```

Exercício 3

Construa um programa que manipula um vetor de inteiros (suponha tamanho do vetor igual a 5). Os números iniciais do vetor devem ser digitados pelo usuário. Posteriormente, o programa manipula esses dados de forma que todo número par deve ser dividido por 2. Ao final, imprimir o vetor resultante.

Solução 2: Uso de uma função manipula_pares, que recebe o vetor e chama uma segunda função manipula_um_par que recebe um número que é par e divide por 2;

```
void manipula_pares(int *vetor, int N); // Protótipo não é modificado
void manipula_um_par(int * par);   // Função incluída nessa versão

int main()
{
   int N=5, i;
   int numeros[5];


   for (i=0;i<N;i++) {
      printf("\nDigite o %do. numero:", i+1);
      scanf("%d",&numeros[i]);
   }

   manipula_pares(numeros,N);   // Chamada não é alterada

   for (i=0;i<N;i++) {
      printf("\nO %do. numero do vetor resultante eh: %d",i+1,numeros[i]);
   }

   return 0;
}

void manipula_pares(int *vetor, int N) {
int i;

   for (i=0;i<N;i++) {
      if (vetor[i] / 2 == vetor[i]/2.0) {
         manipula_um_par(&vetor[i]);    //Chama a nova função para alterar o conteúdo de uma posição do vetor
      }
   }
}

void manipula_um_par(int *par)  // Função incluída. Recebe o endereço do inteiro a ser alterado (observe  a chamada dessa função)
{
   *par=*par/2;      //Mesmo o valor sendo um elemento de um vetor, não precisa do índice pois está alterando o valor do inteiro diretamente.
}
```

Exercício 4

Construa um programa que manipula um vetor de struct dados (suponha tamanho do vetor igual a 5). A struct possui um campo nome e outro campo que armazena um inteiro. Os números e nomes devem ser digitados pelo usuário. Posteriormente, o programa manipula esses dados de forma que todo número par dentro de uma struct deve ser dividido por 2. Ao final, imprimir o vetor resultante.

Solução 1: a função que manipula 1 par, recebe o valor inteiro.

```
struct dados_estruturados{
  char nome[20];
   int numero;
};
typedef struct dados_estruturados dados;

void manipula_pares(dados *vetor, int N);   //Aletra o protótipo para passar vetor de struct
void manipula_um_par(int * par);


int main()
{
   int N=5, i;
   dados vetor[5];


   for (i=0;i<N;i++) {
      printf("\nDigite o %do. numero:", i+1);
      scanf("%d",&vetor[i].numero);
      printf("\n\n Entre com o %do. nome: ",i+1);
      setbuf(stdin,NULL);
      scanf("%s",vetor[i].nome);

   }
   manipula_pares(vetor,N);  // Apenas o protótipo foi modificado. A chamada permanece a mesma.
   for (i=0;i<N;i++) {
      printf("\nO %do. numero do vetor resultante eh: %d",i+1,vetor[i].numero);
   }

   return 0;
}
```

```
void manipula_pares(dados *vetor, int N) {
int i;

   for (i=0;i<N;i++) {
      if ((vetor[i].numero) / 2 == (vetor[i].numero)/2.0) {
         manipula_um_par(&vetor[i].numero);
      }                                  //Precisa acessar o campo número da struct do tipo dados
   }
}



  // 1a versao: passa o campo numero diretamente

  void manipula_um_par(int *par)  // Não modifica, uma vez que passa o endereço de um inteiro
  {
     *par=*par/2;
  }
```

1ª Versao: uso de vetor de inteiros

```
void manipula_pares(int *vetor, int N) {
int i;

   for (i=0;i<N;i++) {
      if (vetor[i] / 2 == vetor[i]/2.0) {
         manipula_um_par(&vetor[i]);
      }
   }
}
```

2ª Versao: uso de vetor de struct, onde cada struct tem um campo inteiro

```
void manipula_pares(dados *vetor, int N) {
int i;

   for (i=0;i<N;i++) {
      if ((vetor[i].numero) / 2 == (vetor[i].numero)/2.0) {
         manipula_um_par(&vetor[i].numero);
      }
   }
}
```

Exercício 5

Construa um programa que manipula um vetor de struct dados (suponha tamanho do vetor igual a 5). A struct possui um campo nome e outro campo que armazena um inteiro. Os números e nomes devem ser digitados pelo usuário. Posteriormente, o programa manipula esses dados de forma que todo número par dentro de uma struct deve ser dividido por 2. Ao final, imprimir o vetor resultante.

Solução 2: a função que manipula 1 número par, deve receber a struct correspondente.

```
struct dados_estruturados{
   char nome[20];
   int numero; };
typedef struct dados_estruturados dados;

void manipula_pares(dados *vetor, int N);   // Não se modifica
void manipula_um_par(dados * par);   // O protótipo foi alterado para receber uma struct e não um valor inteiro

int main()
{
   int i, N=5;
   dados vetor[5];

   for (i=0;i<N;i++) {
      printf("\nDigite o %do. numero:", i+1);
      scanf("%d",&vetor[i].numero);
      printf("\nEntre com o %do. nome: ",i+1);
      setbuf(stdin,NULL);
      scanf("%s",vetor[i].nome);

   }
   manipula_pares(vetor,N);  // Não se modifica

for (i=0;i<N;i++) {
      printf("\nO %do. nome do vetor resultante eh: %s",i+1,vetor[i].nome);
      printf("\nO %do. numero do vetor resultante eh: %d",i+1,vetor[i].numero);

   }

   return 0;
}
```

```
void manipula_pares(dados *vetor, int N) {
int i;

   for (i=0;i<N;i++) {
      if ((vetor[i].numero) / 2 == (vetor[i].numero)/2.0) {
         manipula_um_par(&vetor[i]);       // Altera a chamada da função para passar o endereço da struct
                                           // e não apenas o inteiro armazenado no campo num
      }
   }
}

// 2a versao: passa o endereço da struct dados e deve manipular o campo numero

void manipula_um_par(dados *par)  //recebe o endereço da struct (1 elemento do vetor)
{
   (*par).numero=(*par).numero/2;
}
```

```
void manipula_pares(dados *vetor, int N) {
int i;

   for (i=0;i<N;i++) {
      if ((vetor[i].numero) / 2 == (vetor[i].numero)/2.0) {
         manipula_um_par(&vetor[i]);       // Altera a chamada da função para passar o endereço da struct
                                           // e não apenas o inteiro armazenado no campo num
      }
   }
}

// 2a versao: passa o endereço da struct dados e deve manipular o campo numero

void manipula_um_par(dados *par)  //recebe o endereço da struct (1 elemento do vetor)
{
    par->numero=par->numero/2;      // Alternativa de sintaxe
}
```

Exercício 6

Construa um programa que manipula um vetor de struct dados (alocado dinamicamente). A struct dados possui um campo nome e outro campo que armazena um inteiro. O número de elementos do vetor deve ser informado pelo usuário. Os números e nomes de cada estrutura devem ser digitados pelo usuário. Posteriormente, o programa manipula esses dados de forma que todo número par dentro de uma struct deve ser dividido por 2. A string nome permanece inalterada. Ao final, imprimir o vetor resultante.

Obs1: A alocação das estruturas e a leitura dos dados deve ser implementado na própria main(). Ao final, liberar o espaço do vetor.

Obs 2: Implementar a função que manipula o vetor na busca dos números pares e a função que manipula 1 número par (essa deve receber a struct correspondente).

Exercício 7 (versão UPM)

Construa um programa que manipula um vetor de struct dados (alocado dinamicamente). A struct dados possui um campo nome e outro campo que armazena um inteiro. O número de elementos do vetor deve ser informado pelo usuário. Os números e nomes de cada estrutura devem ser digitados pelo usuário. Posteriormente, o programa manipula esses dados de forma que todo número par dentro de uma struct deve ser dividido por 2. A string nome permanece inalterada; Ao final, imprimir o vetor resultante.

Obs1: A alocação das estruturas e a leitura dos dados devem ser implementados em uma função auxiliar aloca_structs(). Atenção: **vetor

Obs 2: Implementar a função que manipula o vetor na busca dos números pares e a função que manipula 1 número par (essa deve receber a struct correspondente).