Avaliação Final

Universidade de Itaúna

Curso: Ciência da Computação

Disciplina: Algoritmos e Estrutura de Dados II

Professor: Adriano Benigno Moreira

Aluno (a): Davi Ventura Cardoso Perdigão Data: 09/12//2020

1) A) Uma struct é uma variável especial que contém diversas outras variáveis normalmente de tipos diferentes. Por exemplo, se for preciso armazenar a altura, o peso e a idade de uma pessoa, pode-se criar uma struct chamada Pessoa e agrupar os dados em um único tipo de dado.

B) Ponteiros ou apontadores, são variáveis que armazenam o endereço de memória de outras variáveis. Dizemos que um ponteiro “aponta” para uma variável quando contém o endereço da mesma. Os ponteiros podem apontar para qualquer tipo de variável. Portanto temos ponteiros para int, float, double, etc. Ponteiros são muito úteis quando uma variável tem que ser acessada em diferentes partes de um programa. Neste caso, o código pode ter vários ponteiros espalhados por diversas partes do programa, “apontando” para a variável que contém o dado desejado. Caso este dado seja alterado, não há problema algum, pois todas as partes do programa têm um ponteiro que aponta para o endereço onde reside o dado atualizado. Existem várias situações em que ponteiros são úteis, por exemplo:

- Alocação dinâmica de memória.
- Manipulação de arrays.
- Retornar mais de um valor em uma função.
- Referência para listas, pilhas, árvores e grafos.

C)

```
#include <stdio.h>

#include <stdlib.h>

#include <locale.h>


int main(void)

{

	int i, quant;

	printf("\n\nInforme a quantidade de produtos: ");
```

```
scanf("%d",&quant);
setlocale(LC_ALL,"Portuguese");
struct carac
{
int codigo;
int quantidade;
float valor;
}produto[quant];

float total[quant];
printf("\n---------- Cadastro do produto ----------\n\n\n");
for(i=0;i<quant;i++)
        {
        printf("Código do %dº produto: ",i+1);
        scanf("%d",&produto[i].codigo);
        printf("Quantidade do %dº produto: ",i+1);
        scanf("%d",&produto[i].quantidade);
        printf("Valor do %dºproduto: ",i+1);
        scanf("%f",&produto[i].valor);
        total[i]=produto[i].quantidade * produto[i].valor;
        printf("\n\n");
        }

printf("\n---------- Listagem do produto ----------\n\n\n");
for(i=0;i<quant;i++)
        {
        printf("\nCodigo .....: %d", produto[i].codigo);
        printf("\nQuantidade .....: %d", produto[i].quantidade);
        printf("\nValor .....: %.2f", produto[i].valor);
        printf("\nTotal a Pagar.....: %.2f\n\n", total[i]);
        }
```

```
    system("pause");

    return(0);

}
```

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

int main(void)
{
    int i, quant;
    printf("\n\nInforme a quantidade de produtos: ");
    scanf("%d",&quant);
    setlocale(LC_ALL,"Portuguese");
    struct carac
    {
        int codigo;
        int quantidade;
        float valor;
    }produto[quant];

    float total[quant];
    printf("\n---------- Cadastro do produto ----------\n\n\n");
    for(i=0;i<quant;i++)
        {
        printf("Código do %dº produto: ",i+1);
        scanf("%d",&produto[i].codigo);
        printf("Quantidade do %dº produto: ",i+1);
        scanf("%d",&produto[i].quantidade);
        printf("Valor do %dºproduto: ",i+1);
        scanf("%f",&produto[i].valor);
        total[i]=produto[i].quantidade * produto[i].valor;
        printf("\n\n");
        }

    printf("\n---------- Listagem do produto ----------\n\n\n");
    for(i=0;i<quant;i++)
        {
        printf("\nCodigo ..........: %d", produto[i].codigo);
        printf("\nQuantidade .......: %d", produto[i].quantidade);
        printf("\nValor ............: %.2f", produto[i].valor);
        printf("\nTotal a Pagar.....: %.2f\n\n", total[i]);
        }


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

2) A)

- Manipulação de uma quantidade imprevisível de dados.
- Acesso indireto dos elementos
- O espaço necessário é alocado durante a execução do programa.

B) LISTA: é uma sequência de elementos de mesmo tipo e armazenados na memória e cada elemento é armazenado em uma cédula.

FILA: tem como principal característica o FIF O(First in, first out) o primeiro elemento armazenado é o primeiro que será retirado e, consequentemente, o ultimo armazenado será o último a ser retirado.

PILHA: caracteriza-se pelo fato de que o último elemento a ser armazenado será o primeiro a ser retirado.

C)

```
//Struct
        typedef struct {
        int dados[tamanho];
        int ini;
        int Topo;
        } TPilha;
        //Empilhar
        void TPilha_Insere (TPilha *p, int x) {
        if(TPilha_Cheia(p) ==1){
        printf("\n Erro: Pilha cheia");
 }
        //Desempilhar
        int TPilha_Retira (TPilha *p) {
        int aux;
        if(TPilha_Vazia(p) == 1){
        printf("\n Erro: A Pilha está vazia");
   }
```

3) A) Pode se denominar o método de bolha como uma comparação dois a dois, pois sempre se compara o elemento com seu sucessor, ou seja, a primeira posição com a segunda, na próxima repetição, campara-se o segundo com o terceiro e assim por diante. Sua principal desvantagem é sua lentidão, por percorrer o vetor diversas vezes, sendo assim não é recomentado para aplicações que exigem velocidade.

B) É um algoritmo de comparação que emprega a estratégia de “divisão e conquista”. A ideia básica é dividir o problema de ordenar um conjunto com n itens em dois problemas menores com base em um elemento da lista que foi escolhido para ser o pivô, e assim se compara os outros elementos com o pivô.

O pivô é um elemento da lista que foi escolhido para servir de comparação e assim ordenar os outros tendo ele como base. É de suma relevância para o funcionamento do Quicksort.

C)

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

void bolha_recursiva (int n, int *v);

int main (void)
{
    int i;
    int v[8] = {25,48,37,12,57,86,33,92};//Vetor desorganizado
    bolha_recursiva(8,v);

    printf("Vetor ordenado: ");

    for (i=0; i<8; i++){
        printf("%d ",v[i]);
    }
    return 0;
}
void bolha_recursiva (int n, int *v) //Metodo de bolha
{
    int j, troca = 0;;
    for (j=0; j<n-1; j++)
        if (v[j]>v[j+1])
                                {
            int temp = v[j];
            v[j] = v[j+1];
            v[j+1] = temp;
            troca = 1;
           }
    if(troca != 0)
           {
```

bolha_recursiva(n-1,v);

}

}

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

void bolha_recursiva (int n, int *v);

int main (void)
{
    int i;
    int v[8] = {25,48,37,12,57,86,33,92};//Vetor desorganizado
    bolha_recursiva(8,v);

    printf("Vetor ordenado: ");

    for (i=0; i<8; i++){
        printf("%d ",v[i]);
    }
    return 0;
}
void bolha_recursiva (int n, int *v) //Metodo de bolha
{
    int j, troca = 0;;
    for (j=0; j<n-1; j++)
        if (v[j]>v[j+1])
            {
            int temp = v[j];
            v[j] = v[j+1];
            v[j+1] = temp;
            troca = 1;
            }
    if(troca != 0)
        {
        bolha_recursiva(n-1,v);
        }

}
```

4) A) C(n) = O(n3). Pois, nesse código, temos 3 laços de repetição FOR encadeado e isso faz com que esse encadeamento seja o caso mais relevante.

B) A finalidade da função é armazenar, em uma matriz C, o produto de uma matriz A com outra matriz B somado com a matriz C e após isso apresentar ao usuário os novos valores da matriz C.

c[i][j] + = a[i][l] * b[l][j];

5) C)

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <locale.h>

```
struct cadastro
{
char nome[30], endereco[30];
int idade;
}pessoa[5];

        int main (void)
        {
        setlocale(LC_ALL, "Portuguese");
        int i;
        FILE *arq;
        arq = fopen("cadastro.txt","wb");
        if(arq == NULL)
                {
                printf("Erro na gravação \n");
                system("pause");
                exit(1);
                }
        for(i=0;i<5;i++)
                {
                printf("\n%dº Pessoa\n\n",i+1);
                printf("Informe seu nome: ");
                fflush(stdin);
                gets(pessoa[i].nome);
                fflush(stdin);
                printf("Informe seu endereço: ");
                gets(pessoa[i].endereco);
                fflush(stdin);
                printf("Informe sua idade: ");
                scanf("%d",&pessoa[i].idade);
```

```
        }
        fwrite(&pessoa,sizeof(struct cadastro),5,arq);
        fclose(arq);
        system("pause");
return 0;
```