(/notific

Algoritmos e Programação Estruturada (/alu...

Av2 - Algoritmos e Programação Estruturada

Sua avaliação foi confirmada com sucesso

Informações Adicionais

Período: 31/07/2023 00:00 à 20/11/2023 23:59
Situação: Cadastrado
Tentativas: 2 / 3
Pontuação: 1000
Protocolo: 921114584

Avaliar Material

1) As funções são utilizadas nos programas para acelerar o processo de programação. Muitas vezes precisamos fazer o mesmo tipo de programa que tem a seguinte opção: “leia um vetor de N números”. Para essa atividade, podemos criar uma função que gera automaticamente o vetor, utilizando a função ***rand ().*** A função ***rand ()*** gera números aleatórios e basta então inserir esses números em um vetor. Suponha que seja necessário ordenar o vetor. As linguagens de programação já possuem métodos de ordenação, os mais populares métodos de ordenação são: *Insertion Sort, Selection Sort, Bubble Sort, Comb Sort, Quick Sort, Merge Sort, Heap Sort e Shell Sort.*

Observe atentamente o programa que demonstra o uso das funções para gerar o vetor e para ordenar o vetor com o método *Bubble Sort*:

```
#include<stdio.h>
#include <stdlib.h>
int r[10];

void gerarRandomico(){
int a;
for(a = 0; a < 10; ++a) {
r[a] = rand()%100;
}}

void ordena(){
 int i, j, aux;
for( i=0; i<10; i++ ){
  for( j=i+1; j<10; j++ ){
    if( r[i] > r[j] ){
      aux = r[i];
      r[i] = r[j];
      r[j] = aux;
    }}}}

int main(){
int i;
gerarRandomico();
ordena();
for ( i = 0; i < 10; i++ ) {
printf("\n Vetor[%d] = %d", i, r[i]);}
return 0; }
```

Tomando como referência o contexto apresentado, julgue as afirmativas a seguir em (V) Verdadeiras ou (F) Falsas .

( ) Uma função criada para retornar um valor *char*, o comando *return* somente poderá retornar o valor *char*.

( ) Uma função pode ser chamada quantas vezes forem necessárias para realizar uma tarefa, a única exceção é de que não podemos fazer essa chamada dentro de uma estrutura de repetição *for ( )*, somente poderá ser utilizado o comando *while ( )*.

( ) No programa apresentado, existem duas funções que não utilizam o *return* porque elas são do tipo *void*.

( ) A variável que foi utilizada no comando *for (i = 0; i < 10; i++)* foi declarada como tipo *int* e deveria ser *float* pois, quando usamos vetores os números podem ser maiores.

( ) No programa apresentado foi utilizado duas funções pois, cada função tem um objetivo específico, devemos evitar misturar as funcionalidades das funções.

Assinale a alternativa que apresenta a sequência correta:

Alternativas:

a) V – V – V – V – V.

b) F – V – V – F – V.

c) F – F – V– F – V.

d) V – F – V – F – V. Alternativa assinalada

e) F – F – F – F – F.

2) “Muitos problemas têm a seguinte propriedade: cada instância do problema contém uma instância menor do mesmo problema. Diz-se que esses problemas têm estrutura recursiva. Para resolver tal problema, pode-se aplicar o seguinte método: se a instância em questão for pequena, resolva-a diretamente; senão, reduza-a a uma instância menor do mesmo problema, aplique o método à instância menor, volte à instância original. A aplicação desse método produz um algoritmo recursivo.” (FEOFILOFF, 2017, p. 1). Baseado nesse conceito, avalie as asserções a seguir:

I - Recursividade significa indicar quando um problema maior pode ser dividido em instâncias menores do mesmo problema.

PORQUE

- II - A técnica de recursividade pode substituir o uso de estruturas de repetição tornando o código mais elegante.

A respeito dessas asserções, assinale a alternativa correta:

Alternativas:

a) As asserções I e II são proposições verdadeiras, e a II é uma justificativa da I. Alternativa assinalada

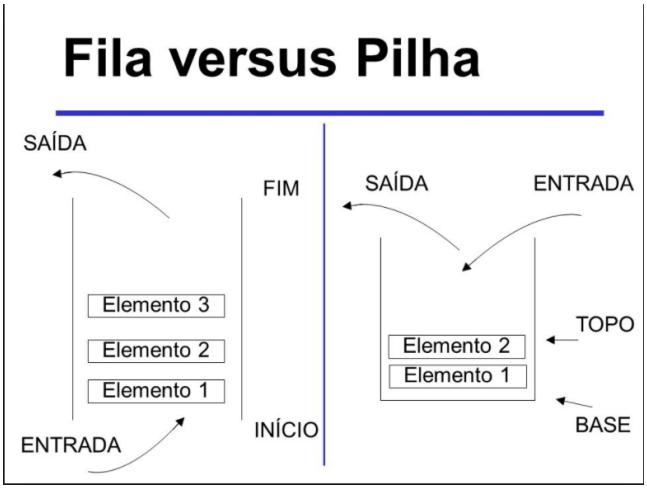
b) As asserções I e II são proposições verdadeiras, mas a II não é uma justificativa da I.

c) A asserção I é uma proposição verdadeira, e a II é uma proposição falsa.

d) A asserção I é uma proposição falsa, e a II é uma proposição verdadeira.

e) As asserções I e II são proposições falsas.

3) Filas e Pilhas são estruturas usualmente implementadas através de listas, restringindo a política de manipulação dos elementos da lista.



Podemos destacar as seguintes características entre Pilhas e Filas:

I. Pilhas e Filas são estruturas de dados com alocação dinâmica de memória, são Listas Encadeadas (ou Ligadas).

II. Ao implementar mecanismos de inserção e remoção de elementos da Lista Encadeada (ou Ligada) pode-se definir se a Lista comporta-se como uma Fila ou como uma Pilha.

III. Tanto uma Pilha como a Fila podem ser implementadas por meio de uma Lista Encadeada ou de um Vetor (Array).

- IV. Enquanto a Fila obedece ao princípio FIFO, uma Pilha é manipulada pelo princípio LIFO.

Análise as afirmativas e escolha a alternativa correta referente as afirmativas:

Alternativas:

a) Somente a afirmativa I está correta.

b) Somente as afirmativas II e III estão corretas.

c) Somente as afirmativas II e IV estão corretas.

d) Somente as afirmativas I, III e IV estão corretas.

e) As afirmativas I, II, III e IV estão corretas. Alternativa assinalada

4) Recursividade é uma técnica sofisticada em programação, na qual uma função chama a si mesma criando várias instâncias (chamadas recursivas). Embora seja uma técnica que proporciona um código mais limpo e facilita a manutenção, seu uso deve levar em consideração a quantidade de memória necessária para a execução do programa. Nesse contexto, avalie as asserções a seguir:

I. A cada chamada recursiva é alocado recursos na memória para a função, se a função for muito grande poderá ocorrer um acúmulo de memória.

PORQUE

- II. É preciso avaliar o custo-benefício em se ter um código mais sofisticado em detrimento de uma estrutura de repetição, pois a segunda opção gasta menos memória.

A respeito dessas asserções, assinale a alternativa correta:

Alternativas:

a) As asserções I e II são proposições verdadeiras, e a II é uma justificativa da I.

b) As asserções I e II são proposições verdadeiras, mas a II não é uma justificativa da I.

c) A asserção I é uma proposição verdadeira, e a II é uma proposição falsa.

d) A asserção I é uma proposição falsa, e a II é uma proposição verdadeira. Alternativa assinalada

e) As asserções I e II são proposições falsas.

5) A seguir é apresentado o código da função "existe(li, item)", a qual verifica se o “item” existe ou não na lista “li”.

```
bool existe(struct Lista* li, int item) {
assert(li != NULL);
struct No* aux = _____;
while(_____){
if(aux->info == item){
return true;
}
_____
}
return false;
}
```

Assinale a alternativa que preenche corretamente as lacunas do código.

Alternativas:

a) NULL / aux != NULL / aux = aux->proximo

b) li->inicio / aux == NULL / aux = aux->proximo

c) NULL / aux == NULL / aux = aux->proximo

d) li->inicio / aux != NULL / aux = aux->proximo Alternativa assinalada

e) li->inicio / aux != NULL / aux = li->proximo