Aap3 - Algoritmos e Programação Estruturada

Informações Adicionais

Período: 15/08/2022 00:00 à 03/12/2022 23:59
Situação: Cadastrado
Tentativas: 2 / 3
Protocolo: 816357294

A atividade está fora do período do cadastro

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. Alternativa assinalada

c) F – F – V– F – V.

d) V – F – V – F – V. ✔

e) F – F – F – F – F.

2) É comum utilizarmos ponteiros com funções. Um caso importante de ponteiros com funções é na alocação de memória dinâmica. A função ***malloc ( )*** pertencente a biblioteca ***<stdlib.h>*** é usada para alocar memória dinamicamente. Entender o tipo de retorno dessa função é muito importante, principalmente para seu avanço, quando você começar a estudar estruturas de dados.

Observe atentamente o programa que demonstra o uso de funções:

```
#include<stdio.h>
#include<stdlib.h>
int* alocar(){
return malloc(200);
}
int main(){
int *memoria;
memoria = alocar();
if(memoria != NULL){
printf("Endereço de memória alocada = %x",memoria);
}
else{
printf("Memória não alocada");
}
return 0;
}
```

Conforme demonstrado no programa apresentado a função *malloc ( )* irá reservar um espaço de memória dinamicamente, a função *malloc ( )* pode retornar dois valores e são eles:

Alternativas:

a) *null* ou um ponteiro genérico (ponteiro genérico é do tipo *void)* ✔

b) *int* ou um ponteiro genérico (ponteiro genérico é do tipo *void)*

c) *float* ou um ponteiro genérico (ponteiro genérico é do tipo *void*)

d) *char* ou numérico (*int ou float*) Alternativa assinalada

e) *string* ou *null*

3)

O escopo é dividido em duas categorias, local ou global. Quando se tem variáveis que ambas são locais, elas existem e são notadas somente dentro do corpo da função onde foram definidas. Para definir uma variável global é preciso criá-la fora da função, assim ela será visível por todas as funções do programa.

Fonte: SCHEFFER, V.C. Escopo e passagem de parâmetros.

- Nesse contexto, analise as asserções a seguir:

I - A utilização de variáveis globais permite dimensionar o uso da memória.

II - As funções e procedimento tem pouca relevância em programação.

III - As variáveis locais são criadas e mantidas na memória ao fim da função

- IV - As variáveis globais permanecem na memória durante todo o tempo de execução.

É correto apenas o que se afirma em:

Alternativas:

a) I e II.

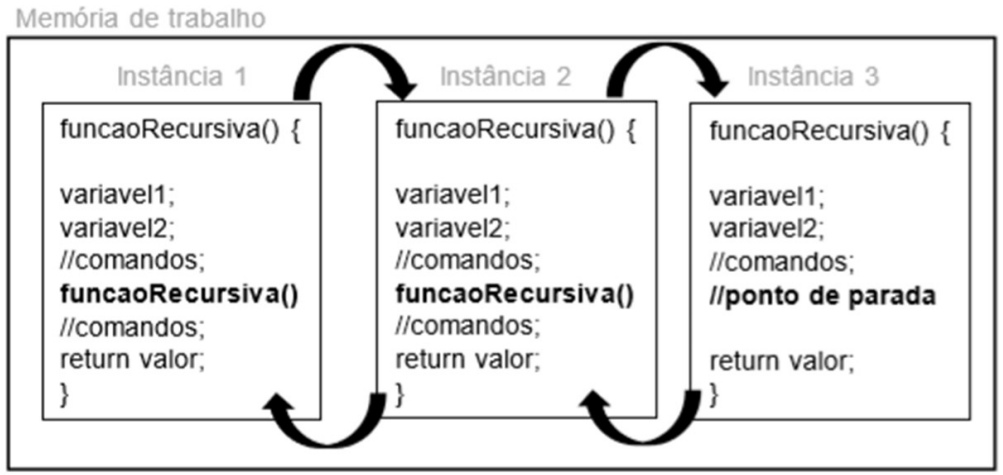
b) III e IV.

c) II e III.

d) I e IV. ✔ Alternativa assinalada

e) I, II, III e IV.

4) Analise a figura a seguir.



Fonte: elaborada pela autora.

Para compreender o mecanismo apresentado na imagem anterior, organize as operações em sequência correta:

1 - Chamada a função funcaoRecursiva(), que por sua vez, possui em seu corpo um comando que invoca a si mesma.

2 - Na terceira instância, uma determinada condição de parada é satisfeita.

3 - Um novo espaço é alocado, com variáveis e comandos. Como a função é recursiva, novamente ela chama a si mesma, criando então a terceira instância da função.

4 - Cada instância da função passa a retornar seus resultados para a instância anterior.

5 - Nesse momento é criada a segunda instância dessa função na memória de trabalho.

Assinale a alternativa que contém a sequência correta:

Alternativas:

a) 1-3-5-2-4.

b) 4-5-3-2-1.

c) 1-5-3-2-4. ✔

d) 1-5-2-4-3.

e) 2-5-1-3-4. Alternativa assinalada