Estácio

Voltar

Teste de Conhecimento

avalie sua aprendizagem

ESTRUTURA DE DADOS
CCT0826_A2_202101110137_V1

Lupa Calc.

Aluno: **DOUGLAS MATOS DA SILVA** Matr.: **202101110137**
Disc.: **ESTRUTURA DE DADOS** **2022.1 EAD (G) / EX**

Prezado (a) Aluno(a),

Você fará agora seu **TESTE DE CONHECIMENTO**! Lembre-se que este exercício é opcional, mas não valerá ponto para sua avaliação. O mesmo será composto de questões de múltipla escolha.

Após responde cada questão, você terá acesso ao gabarito comentado e/ou à explicação da mesma. Aproveite para se familiarizar com este modelo de questões que será usado na sua AV e AVS.

Saiba mais

1. Assinale a opção CORRETA. O protótipo da função que permitirá calcular o volume de um paralelepípedo com medidas **a**, **b** e **c** de acordo com o trecho da chamada : **cout << "Volume = " << volume(a,b,c);**

- ( ) **void volume (float, float, float &);**
- (✔) **float volume (float, float, float);**
- ( ) **float volume (float ; float ; float);**
- ( ) **float volume (float, float);**
- ( ) **void volume (float , float);**

Explicação:

Resposta única.

Pela chamada da função

cout << "Volume = " << volume(a,b,c);

temos que a função retorna um valor, que pelo enunciado, vemos que é do tipo float. Há ainda os valores de a, b e c que são medidas de uma figura geométrica. Logo, a, b e c são do tipo float.

Como protótipo é uma declaração temos :

tipo de retorno da função seguido do seu **nome** seguido de **parênteses** e dentro dos parênteses, temos os **tipos das variáveis que estavam na chamada da função**.

Logo : **float volume(float, float, float);**

Note a vírgula para separar os tipos e o ponto e vírgula ao final.

Saiba mais

2. O que será impresso pela função Eureka ? Assinale a opção correta.

```
void Eureka()
 {
        for (int i = 1; i <= 10; i++)
                if (i % 2 == 0)
                        cout << i << " ";
                else
                  if (i % 7 == 0)
                    return;
 }
```

- (✔) 2 4 6
- ( ) 2 4 6 8 10
- ( ) Nada é impresso, pois a função não compila. A função não deveria usar **return**, pois não está de acordo com o uso de **void**.
- ( ) Nada é impresso, pois a função não compila. Para a função compilar, deveria ter **int** no lugar de **void**, já que há **return** na função.
- ( ) 2 4 6 8

Saiba mais

3. Qual será a saída para o seguinte trecho de código?

```
void FUNC1()
{
int B = -100;
cout << "Valor de B dentro da função FUNC1: " << B;
}
void FUNC2() {
int B =  -200;
cout << "Valor de B dentro da função FUNC2: " << B;
}
int  main() {
int B = 10;
cout << "Valor de B: " << B;
B = 20;
FUNC1();
cout << "Valor de B: " << B;
B = 30;
FUNC2();
cout << "Valor de B: " << B;
}
```

- ( ) Valor de B: 10 Valor de B dentro da função FUNC1: -100 Valor de B: 20 Valor de B dentro da função FUNC2: -200 Valor de B: 20
- ( ) Valor de B: 10 Valor de B dentro da função FUNC1: -200 Valor de B: 20 Valor de B dentro da função FUNC2: -200 Valor de B: 30
- ( ) Valor de B: 10 Valor de B dentro da função FUNC1: -100 Valor de B: 10 Valor de B dentro da função FUNC2: -200 Valor de B: 30
- (✔) Valor de B: 10 Valor de B dentro da função FUNC1: -100 Valor de B: 20 Valor de B dentro da função FUNC2: -200 Valor de B: 30
- ( ) Valor de B: 10 Valor de B dentro da função FUNC1: -100 Valor de B: 20 Valor de B dentro da função FUNC2: -400 Valor de B: 30

Explicação:

A execução começa pela main. Na main, B recebe 10. E daí é impresso : **Valor de B : 10**

DEpois, B muda e recebe 20.

Em seguida, é chamada a função FUNC1. Dentro desta função, existe um outro B que recebe -100. Então, é impresso :

Valor de B dentro da função FUNC1: -100

A função termina e voltamos para a main, onde existe um outro cout que imprime o valor de B, que na main vale 20. Então, será impresso na tela :

Valor de B : 20

Em seguida, ainda na main, B recebe 30 e logo depois, FUNC2 é chamada. Executando FUNC2 temos que um B, local à função, recebe -200 e logo em seguida é impresso : **Valor de B dentro da função FUNC2 : -200**

Terminando a execução de FUNC2, voltamos para a main onde é impresso o valor do B (local à main) : **Valor de B : 30**

Logo, a opção correta é a que diz :

Valor de B : 10
Valor de B dentro da função FUNC1: -100
Valor de B : 20
Valor de B dentro da função FUNC2 : -200
Valor de B : 30

Saiba mais

4. Diga, para o trecho do programa abaixo, qual a opção que representa a saída em tela. Considere que o programa será compilado sem erros e irá executar também sem problemas .

```
void troca (int x, int y){
  int tmp;
  tmp = y;
  y = x;
  x = tmp;
```

```
}

int main( ) {
  int a = 13, b = 10;
  troca( a, b );
  cout<<"Valores: "<< a<<"\t"<< b<<< endl;
  system("pause");
}
```

- ( ) Valores: 13 13
- ( ) Valores: 10 13
- ( ) Valores: 31 01
- (✔) Valores: 13 10
- ( ) Valores: 10 10

Explicação:

Acompanhando passo a passo a execução do trecho dado temos :

```
void troca (int x, int y){
  int tmp;
  tmp = y;
  y = x;
  x = tmp;
}

int main( ) {
  int a = 13, b = 10;
  troca( a, b );
  cout<<"Valores: "<< a<<"\t"<< b<<< endl;
  system("pause");
}
```

A execução começa pela main e os valores 13 e 10 são passados para a função na chamada em **troca(a,b);**

Ao iniciar a execução em troca, temos que x recebe 13 e y recebe 10. Mas x e y são parâmetros passados por valor então, nada ocorrerá com a e b na main. ÃSsim, após a função terminar sua execução e voltarmos para a main temos a seguinte impressão na tela :

Valores : 13 10

Saiba mais

5. O que será mostrado na tela pelo programa abaixo ?

```
#include < iostream >
using namespace std;

int a,b;

void dobro(int x){    //x passado por valor
     x=2*x;
}
int triplo(int y){   //y passado por valor
   return 3*y;
}
void altera(int x, int &y)  {   //x passado por valor e  y passado por referencia
     x=x+a;
     y=x+b;
}

int main (){
     a=2;
     b=3;
     dobro(a);
     b=triplo(b);
     altera(a,b);
     cout<< a << " e " << b << endl;

}
```

- ( ) 4 e 9
- ( ) 9 e 16
- ( ) 2 e 9
- (✔) 2 e 13
- ( ) 4 e 12

Explicação:

Executando o programa, passo a passo, temos :

Na main, a recebe 2 e b recebe 3. Note que a e b são variáveis globais.

Daí, ainda na main, a função dobro é chamada. Ao executar dobro, temos que o valor 2 é passado e x (local à dobro) recebe 2 * 2, que dá 4.

Ao terminar a função dobro, volta-se para a main, mas o valor de a passado não mudou, continuando 2.

Após a execução de dobro, a função triplo é chamada na main e é passado o valor de b, que é 3. Iniciando a execução da função triplo, temos que y recebe 3 e a função retorna 3 * 3, que é 9. Voltando para a main, temos que b recebe o valor retornado pela função, que é 9.

Até este ponto, temos que a vale 2 e b mudou para 9.

Em seguida, a função altera é chamada e são passados 2 e 9, respectivamente, para x e y. Executando a função altera, temos que x recebe 2+2, que dá 4 e y recebe 4+9, que dá 13. Como x é passado por valor e y por referência, temos que a mudará e b mudará (ambas na main).

Então, voltando para main temos que a vale 2 e que b vale 13. Assim, será impresso na tela

```
2 e  13
```

Saiba mais

6. Funções são semelhantes aos procedimentos, exceto que uma função sempre retorna um valor. Um exemplo de função seria o conjunto de instruções para calcular o fatorial de um número e após a função ser executada, ela deve retornar o fatorial do número pedido. Marque a opção que representa um protótipo de função válido.

- (✔) retorno nomeFuncao(parametros);
- ( ) nome tipo(parametros);
- ( ) void float(int a, int b);
- ( ) tipo parametros(int a, int b);
- ( ) tipo parametros(parametros);

Explicação:

Por definição, o protótipo de uma função é formado da seguinte forma :

```
nome_da_função ( );
```

Saiba mais

7. Considere a função abaixo:

```
void func (int a, int &b) {
  a++;
  b = a*2;
}
```

e o seguinte trecho de código na função main :

```
int x=2, y=3;
func (x,y);
func (y,x);
cout << x << " ; " << y;
```

Após a execução do cout o que será impresso ?

- (✔) 14; 6
- ( ) 2; 6
- ( ) 2; 12
- ( ) 4; 12
- ( ) 6; 14

Explicação:

Fazendo um teste de mesa, chegaremos na opção correta. Iniciando a execução pela main, temos que x recebe 2 e y recebe 3.

Em seguida, na 1ª. chamada de func, os valores de x e de y, respectivamente, 2 e 3, são passados para a função.

Executando a função func, que possui o parâmetro a passado por valor e o parâmetro b passado por referência (usa &) ...

```
void func (int a, int &b) {
a++;
b = a*2;
```

}

Temos que a recebeu 2 e b recebeu 3. Daí, a ficará 3 e b receberá 6.

Assim, ao terminar a função func e retornarmos para a main teremos x sem alteração igual a 2 e y com alteração igual a 6.

Após voltarmos para a main, temos uma nova chamada de func que recebe os valores de y e de x, respectivamente, que são 6 e 2. Na 2ª. chamada de func para esses valores, teremos que a recebe 6 e y recebe 2. Dentro da função, temos que a é incrementado para 7 e que b recebe 7 * 2 que dá 14. Ao terminarmos a execução da função, temos que y não sofreu mudança, ficando com o valor 6 e que x mudou para 14. Logo, será impresso 14;6

Logo, as demais opções são incorretas.

Saiba mais

8. Caso uma estrutura homogênea (vetor) seja passada como parâmetro para uma função, então:

- (●) Essa passagem é "por referência"
- ( ) Essa passagem pode ser "por valor" ou "por referência"
- ( ) Essa passagem é "por valor"
- ( ) Haverá um erro de compilação, pois vetores não podem ser parâmetros de funções
- ( ) Todos os valores contidos no vetor são copiados para a função

Explicação:

Quando o vetor é um parâmetro de uma função ele é sempre passado por referência, não havendo outra possibilidade. Não ocorrerá erro, se o vetor for devidamente passado para a função.