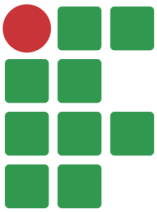

INSTITUTO FEDERAL DA PARAÍBA
CAMPUS CAMPINA GRANDE
BACHARELADO EM ENGENHARIA DA COMPUTAÇÃO
DISCIPLINA DE POO e LAB. POO
PROF. VICTOR ANDRÉ PINHO DE OLIVEIRA

Atividade Unidade I - 1 - Fundamentos C++

Instruções

Responda às questões abaixo diretamente neste documento. As questões valem 2 pontos, cada.

Questões

1. Qual é a diferença, se é que existe, entre as seguintes definições:

   int month = 10, day = 15;

   int month = 010, day = 015;

R: Ao imprimir as variáveis de int month = 10, day = 15, vou obter respectivamente 10 e 15. Caso eu imprima as variáveis int month = 010, day = 015, vou obter 8 e 13, isso porque eles estarão em forma de números octais.

2. Qual a diferença entre definição e declaração de uma variável em C++?

R: Definição seria a criação da variável ou objeto. A Declaração comunica a existência da variável/objeto, irei utilizar múltiplas vezes ao longo do código.

3. Usando a biblioteca iostream da C++, escreva um programa que leia o primeiro nome e 3 notas de um aluno e em seguida exiba na tela uma mensagem como “O aluno FULANO obteve média MÉDIA”. Considere a média ponderada, sendo os pesos 3, 4 e 5, respectivamente, das notas 1, 2 e 3.

```cpp
#include <iostream>

using namespace std;

int main() {

 string nome;

 double n1 , n2, n3, media;

 cin >> nome >> n1 >> n2 >> n3;

 media = ((n1 * 3) + (n2 * 4) + (n3 * 5)) / 12;

 cout << "O aluno " << nome << " obteve média " << media << endl;

 return 0;

}
```

4. Vimos que a E/S da C++ se dá por meio dos objetos cout e cin. Para enviar um fluxo de caracteres para a saída usamos o operador << com cout e, para receber algum dado da entrada padrão, usamos >> com cin. Originalmente, tais operadores são usados no contexto de números inteiros para realizar operações a nível de bit (bitwise), mas no contexto de E/S com cout e cin eles mudam de comportamento. Como se chama essa técnica/recurso? **R: sobrecarga de operadores.**

5. Considere o código abaixo e responda:

```cpp
int i = 42;
int main()
{
   int i = 100;
   int j = ::i; // Colocando :: antes da variável i
}
```

   a. Existe algum problema (de compilação) com o código acima? Por quê? **R: O único problema a princípio seria a variável j que não está sendo usada no código.**

   b. Da forma como o código se encontra, qual será o valor de j? **R: 100**

   c. Considerando que temos duas variáveis i, que modificação seria necessária no código para a variável j ser inicializada com o valor da outra variável i. **R:** `Colocando :: antes da variável i`

6. O que é uma Referência em C++? **R: Seria apenas uma outra variável com outro nome recebendo uma variável que já existe.**

7. Qual a saída do código abaixo? Explique.

```
#include <iostream>
int main()
{
   int i, &ri = i;
   i = 5; ri = 10;
   std::cout << i << " " << ri << std::endl;
   return 0;
}

Saída:
 10 10

o &ri é uma referência para a variável i.
Mesmo com i = 5, o ri recebeu 10, logo tanto o ri quanto i irão ter o
valor 10, pois a partir do momento que &ri virou uma referência para i,
o que mudar em ri irá mudar em i;
```

8. Qual é a diferença entre Ponteiros e Referências?

   R: Ponteiros irão receber o endereço de memória de alguma variável;

   Referência seria apenas uma nova variável com o mesmo valor.

9. O que faz o seguinte trecho de código?

```
int i = 10; // criação de uma nova variável com valor 10
int *pi = &i; // um ponteiro que recebe o endereço de memória de i
*pi = *pi * *pi;
/*
o ponteiro recebe a multiplicação entre os valores
armazenados em pi (10), resultando em 100.
*/
```

10. Quais das seguintes inicializações estão corretas? Quais estão incorretas? Indique o motivo em cada caso.

    a. int i = -1, &r = 0; **R: Errado, referências não podem receber valores, mas sim outras variáveis.**

    b. const int i2 = i, &r = i; //considere o i definido na letra a. **R: Certo, &r e i estão recebendo uma variável, isso é permitido.**

c. int *const p2 = &i2; //considere o i2 definido na letra b. **R: Errado, pois o p2 é um ponteiro constante do tipo int, não pode receber uma referência do tipo const int, já que const é apenas o ponteiro.**

d. const int j = -1, &s = 0; **R: Certo, o 0 é uma constante literal, então devo usar o const antes do int para dar certo.**

e. const int *p1 = &i2; //considere o i2 definido na letra b. **R: Certo, um ponteiro recebendo o endereço de memória de i2.**

f. const int *const p3 = &i2; //considere o i2 definido na letra b. **R: Certo, agora o ponteiro está mais restrito que o ponteiro da letra e.**

g. const int &const r2; **R: Errado, não podemos colocar o tipo const depois do &.**