Curso: C++ Moderno
Período: 1º semestre/2017
Aluno: Arthur Nunes de Paiva Santos Queiroz

Lista de exercícios Módulo 03: Objetos: armazenamento, inicialização

Questões

Questão 1

lmplemente versoes de secret( ), uma para cada caso estudado, de forma que a execução do programa resulte em um erro. Considere a definição de Id abaixo.

p1.cpp

```
void secret(Id* id){
        delete id;
}
```

Questão 2

Verifique, em sua plataforma, qual é a ordem de inicialização de objetos estáticos não-locais.

Em minha plataforma (gcc version 4.8.4 (Ubuntu 4.8.4-2ubuntu1~14.04.3), o objeto Entropy é construido antes de RandNum.

Questão 3

Qual é a ordem de destruição de objetos automáticos? Por quê é assim?

Eles são destruídos na ordem reversa em que foram declarados. Caso não fosse assim, não seria possível garantir que os destrutores de objetos da pilha que dependem em objetos declarados anteriormente seriam executados com dependências válidas.

Fonte: http://stackoverflow.com/a/1245865/702828

Questão 4

Faça com que extendTemporary( ) compile corretamente com a adição de um token.

Este exercício não foi realizado.

Questão 5

lnspecione a implementação do operator new e operator delete de sua plataforma.

Os operadores "new" e "delete" utilizam "malloc" e "free".

Fonte: https://gcc.gnu.org/viewcvs/gcc/trunk/libstdc%2B%2B-v3/libsupc%2B%2B/new_op.cc?view=markup

https://gcc.gnu.org/viewcvs/gcc/trunk/libstdc%2B%2B-v3/libsupc%2B%2B/del_op.cc?view=markup

Questão 6

O que significa a sintaxe Value v( )? Pesquisa sobre a questão na Internet.

"Value v();" representa um caso de "vexing parse" em C++. A expressão aparentemente possui duas interpretações:

- A definição de uma varíavel "v" do tipo Value;
- A declaração de uma função "v", sem parâmetros, que retorna um objeto do tipo Value.

O padrão C++ define a expressão em questão sempre deve ser interpretada da segunda forma.

Fontes: http://stackoverflow.com/q/1424510/702828
https://en.wikipedia.org/wiki/Most_vexing_parse

Questão 7

Como os objetos Person e PersonPOD são inicializados em C++11 (quando não há erro)?

```
PersonPOD a;          // Variáveis inicializadas com lixo.
PersonPOD b();        // Compilador interpreta como declaração de função e acusa erro.
PersonPOD c{};        // Variáveis initializadas com zero.
Person d;             // Compilador reclama que não há construtor default para a classe.
Person e("eu");       // Inicializa objeto corretamente.
PersonPOD f("eu");    // Compilador reclama que nenhum construtor padrão foi encontrado.
Person g{"eu"};       // Inicializa objeto corretamente.
PersonPOD h{"eu"};    // Inicializa objeto corretamente.
Person i("eu", 30);   // Inicializa objeto corretamente.
Person j{"eu", 30};   // Inicializa objeto corretamente.
PersonPOD k("eu", 30); // Compilador reclama que não há construtores viáveis.
PersonPOd l{"eu", 30}; // Objeto é inicializado corretamente.
```