

C++ - Módulo 07

C++ modelos

Summary:

Este documento contém as lições do Módulo 01 dos módulos Ce módulos.

Versão: 6

Conteúdo

| | | |
|------------|---|----------|
| I | Introdução | 2 |
| II | Regras gerais | 3 |
| III | Exercício 00: Comece com algumas funções | 5 |
| IV | Exercício 01: Iter | 7 |
| V | Exercício 02: Matriz | 8 |

Capítulo I

Introdução

CA é uma linguagem de programação de uso geral criada por B. Jarne Stroustrup como uma ex-alternativa à linguagem de programação C, ou "C with Classes" (fonte: Wtk(ped(a .

O objetivo desses módulos é apresentá-lo à programação orientada a objetos. Esse será o ponto de partida de sua jornada em C++. Muitas linguagens são recomendadas para aprender OOP. Decidimos escolher o C++, pois ele é derivado do seu velho amigo C. Como essa é uma linguagem complexa, e para manter as coisas simples, seu código obedecerá ao padrão C1+98.

Sabemos que o C++ moderno é muito diferente em vários aspectos. Portanto, se você quiser se tornar um desenvolvedor C++ proficiente, cabe a você ir além do 42 Common Core!

Capítulo II

Regras gerais

Compilação

- Compile seu código com `c++` e os sinalizadores `-Wall -text ra-terror`
- Seu código ainda deverá ser compilado se você adicionar o flag `-std=c++98`

Convenções de formatação e nomenclatura

- Os diretórios de exercícios serão nomeados da seguinte forma: `ex00`, `ex01`, ..., `exn`
- Nomeie seus arquivos, classes, funções, funções de membro e atributos conforme exigido nas diretrizes.
- Escreva os nomes das classes no formato `UpperCamelCase`. Os arquivos que contêm código de classe sempre serão nomeados de acordo com o nome da classe. Por exemplo:
`ClassName .hpp` / `ClassName .h`, `ClassName .cpp` ou `ClassName .App`. Então, se você tiver um arquivo de cabeçalho que contenha a definição de uma classe "BrickWall" que representa uma parede de tijolos, seu nome será `BrickWall .hpp`.
- A menos que especificado de outra forma, todas as mensagens de saída devem ser encerradas com um caractere de nova linha e exibidas na saída padrão.
- *Adeus Norminelle!* Nenhum estilo de codificação é imposto nos módulos CH. Você pode seguir seu estilo favorito. Mas lembre-se de que um código que seus colegas avaliadores não conseguem entender é um código que eles não podem avaliar. Faça o possível para escrever um código limpo e legível.

Permitido/Proibido

Você não está mais codificando em C. É hora de usar o C1+! Portanto:

- Você tem permissão para usar quase tudo da biblioteca padrão. Portanto, em vez de se ater ao que já sabe, seria inteligente usar o máximo possível as versões em C++ das funções em C com as quais está acostumado.
- No entanto, você não pode usar nenhuma outra biblioteca externa. Isso significa que as bibliotecas C++11 (e formas derivadas) e Boost são proibidas. As seguintes funções também são proibidas: `printf()`, `alloc()` e `free()`. Se você as usar, sua nota será 0 e pronto.

- Observe que, a menos que explicitamente declarado de outra forma, o uso do namespace `4ns_nane` e das palavras-chave `friend` é proibido. Caso contrário, sua nota será -42.
- Você tem permissão para usar o STL somente nos Módulos 08 e 09. Isso significa: nada de contêineres (vetor/lista/mapa/e assim por diante) e nada de algoritmos (qualquer coisa que exija a inclusão do cabeçalho `algor it hut`) até lá. Caso contrário, sua nota será -42.

Alguns requisitos de design

- O vazamento de memória também ocorre em C++. Quando você aloca memória (usando a palavra-chave `new`), deve evitar vazamentos de memória.
- Do Módulo 02 ao Módulo 09, suas aulas devem ser elaboradas na Forma Canônica Ortodoxa, exceto quando explicitamente indicado de outra forma.
- Qualquer implementação de função colocada em um arquivo de cabeçalho (exceto para modelos de função) significa 0 para o exercício.
- Você deve ser capaz de usar cada um dos seus cabeçalhos independentemente dos outros. Portanto, eles devem incluir todas as dependências de que precisam. No entanto, você deve evitar o problema da inclusão dupla adicionando proteções de inclusão. Caso contrário, sua nota será 0.

Leia-me

- Você pode acrescentar alguns arquivos adicionais, se necessário (ou seja, para dividir seu código). Como essas tarefas não são verificadas por um programa, sinta-se à vontade para fazer isso, desde que entregue os arquivos obrigatórios.
- Às vezes, as diretrizes de um exercício parecem curtas, mas os exemplos podem mostrar requisitos que não estão explicitamente escritos nas instruções.
- Leia cada módulo completamente antes de começar! De fato, faça isso.
- Por Odin, por Thor! Use seu cérebro!!!




Você terá de implementar muitas classes. Isso pode parecer entediante, a menos que você consiga criar scripts em seu editor de texto favorito.



Você tem certa liberdade para concluir os exercícios. Entretanto, siga as regras obrigatórias e não seja preguiçoso. Você perderia muitas informações úteis! Não hesite em ler sobre conceitos teóricos.

Capítulo III

Exercício 00: Comece com alguns funções

| | |
|---|---------------------------------|
|  | Exercício : 00 |
| Comece com algumas funções | |
| Diretório de entrada : ez00/ | |
| Arquivos para entregar: | , principal . Qualquer que hpp} |
| Arquivo de maquiagem | cpp , seja . (h , |
| Funções proibidas : Nenhuma | |

Implemente os seguintes modelos de função:

- swap: Troca os valores de dois argumentos fornecidos. Não retorna nada.
- min: compara os dois valores passados em seus argumentos e retorna o menor deles. Se os dois forem iguais, ele retornará o segundo.
- max: Compara os dois valores passados em seus argumentos e retorna o maior deles. Se os dois forem iguais, ele retornará o segundo.

Essas funções podem ser chamadas com qualquer tipo de argumento. O único requisito é que os dois argumentos tenham o mesmo tipo e sejam compatíveis com todos os operadores de comparação.



Os modelos devem ser definidos nos arquivos de cabeçalho.

Executando o código a seguir:

```
int    main( void ) {

    int a = 2;
    int b = 3;

    ::swap( a, b );
    std::cout << "a = " << a << ", b = " << b << std::endl;
    std::cout << "min( a, b ) = " << ::min( a, b ) << std::endl;
    std::cout << "max( a, b ) = " << ::max( a, b ) << std::endl;

    std::string c = "chaine1";
    std::string d = "chaine2";

    ::swap( c, d );
    std::cout << "c = " << c << ", d = " << d << std::endl;
    std::cout << "min( c, d ) = " << ::min( c, d ) << std::endl;
    std::cout << "max( c, d ) = " << ::max( c, d ) << std::endl;


    return 0;
}
```

Deve produzir:

```
a = 3, b = 2
min(a, b) = 2
max(a, b) = 3
c = chaine2, d = chaine1
min(c, d) = chaine1
max(c, d) = chaine2
```

Capítulo IV

Exercício 01: Iteração

| | |
|---|---------------|
|  | Exercício: 01 |
| Iter | |
| Diretório de entrada: ez01/ | |
| Arquivos a serem entregues : principal . iter .(h, hpp) Makefile , cpp , | |
| Funções proibidas : Nenhuma | |

Implemente um modelo de função iter que recebe 3 parâmetros e não retorna nada.

- O primeiro parâmetro é o endereço de uma matriz.
- O segundo é o comprimento da matriz.
- A terceira é uma função que será chamada em cada elemento da matriz.


Entregue um arquivo main . cpp que contenha seus testes. Forneça código suficiente para gerar um executável de teste.

Seu modelo de função iter deve funcionar com qualquer tipo de matriz. O terceiro parâmetro pode ser um modelo de função instanciado.

Capítulo V

Exercício 02:

Matriz

| | |
|---|---------------|
|  | Exercício: 02 |
| Matriz | |
| Diretório de entrada: ez02/ | |
| Fiestoturnin : Makefile, main.cpp, Array.{h, hppl} e arquivo opcional: Array.tpp | |
| Funções proibidas : Nenhuma | |

Desenvolva um modelo de classe Array que contenha elementos do tipo T e que implemente os seguintes comportamentos e funções:

- Construção sem parâmetro: Cria uma matriz vazia.
- Construção com um int n sem sinal como parâmetro: Cria uma matriz de n elementos inicializados por padrão.
Dica: tente compilar `int * a = new int ();` em seguida, exiba `*a`.
- Construção por cópia e operador de atribuição. Em ambos os casos, a modificação da matriz original ou de sua cópia após a cópia não deve afetar a outra matriz.
- Você DEVE usar o operador `new` para alocar memória. A alocação preventiva (alocalizando a memória com antecedência) é proibida. Seu programa nunca deve acessar a memória não alocada.
- Os elementos podem ser acessados por meio do operador subscripto: `array[i]`.
- Ao acessar um elemento com o operador, se o seu índice estiver fora dos limites, é lançado um `std::exception`.
- Uma função de membro `size()` que retorna o número de elementos na matriz. Essa função de membro não recebe nenhum parâmetro e não deve modificar a instância atual.

Como de costume, certifique-se de que tudo funcione conforme o esperado e entregue um arquivo `.cpp` principal que contenha seus testes.