QF5204 Supplementary Guideline

# C++ in Quant Interviews

*Gang GUO*

# Content

Part1: Question category

Part2: Preparation for different questions

Part3: Recommended reading

# Part1: Question category

**Here are some usual interview question types:**

1. **Terminology explanation**
2. **Common sense of C++ basics**
3. **Output(s) or error(s)s for chunk of codes**
4. **Algorithm questions**
5. **Design pattern of your project (advanced)**

# Part1: Question category

## 1. Terminology explanation

- what's encapsulation
- what's constructor/destructor/copy constructor
- what's inheritance/polymorphism
- what's memory allocation/deallocation
- what's memory leak
- what's the difference of overwrite and overload
- what's virtual function/virtual table
- difference between array & list and how to create them

…

# Part1: Question category

## 2. Common sense of C++ basics

**- Pointer:**

```
int* - pointer to int
int const * - pointer to const int
int * const - const pointer to int
int const * const - const pointer to const int
int ** - pointer to pointer to int
int ** const - a const pointer to a pointer to an int
int * const * - a pointer to a const pointer to an int
int const ** - a pointer to a pointer to a const int
int * const * const - a const pointer to a const pointer to an int
```

**- how to declare a dynamic array**

**- how to handle a constructor that fails**

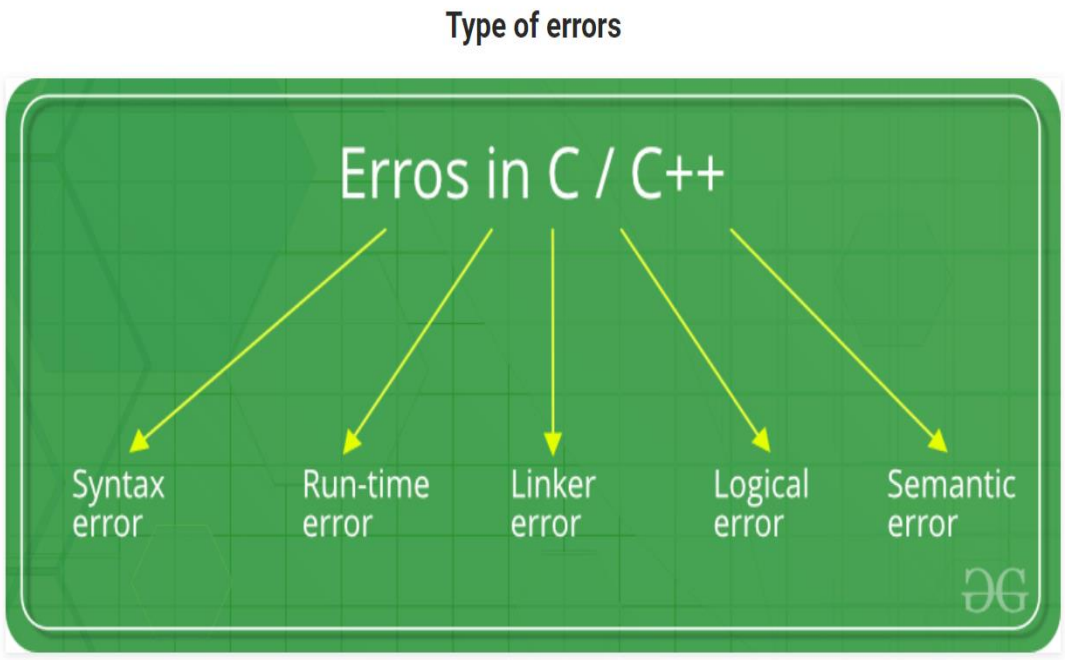
**- what's the order that local objects are destructed**

**…**

# Part1: Question category

## 3. Output(s) or error(s)s for chunk of codes

- **what's the output of codes:**
- **What's the error(s) of code**




```cpp
#include<iostream>

using namespace std;
class Base1 {
 public:
     Base1()
     { cout << " Base1's constructor called" << endl;  }
};

class Base2 {
 public:
     Base2()
     { cout << "Base2's constructor called" << endl;  }
};

class Derived: public Base1, public Base2 {
   public:
     Derived()
     {  cout << "Derived's constructor called" << endl;  }
};

int main()
{
   Derived d;
   return 0;
}
```

Ouput:

Base1's constructor called

Base2's constructor called

Derived's constructor called

# Part1: Question category

## 4. Algorithm questions

**Topics :**

- 1. Graph
- 2. Linked List
- 3. Dynamic Programming
- 4. Sorting And Searching
- 5. Tree / Binary Search Tree
- 6. Number Theory
- 7. BIT Manipulation
- 8. String / Array

# Part1: Question category

## 5. Design Pattern

**Note**: design pattern is not C++ language design pattern, but your derivative pricing/trading framework. Dr. Li Hao has embedded the design pattern into every course

- Common questions:
  1. how could you ensure extendability / flexibility / etc. in your framework?
  2. how do you deal with module interaction
  3. how do you deal with error handling? Using log?
  4. show me your framework
  …

# Part2: Preparation for different questions

**1. Terminology explanation**

**2. Common sense of C++ basics**

- Firstly, read cheat sheet of C++ → grasp the framework and components of this language

  Click here to download cheat sheet;

  Online sources: *Chinese Web* and *English Web1*, *Geeksforgeeks*

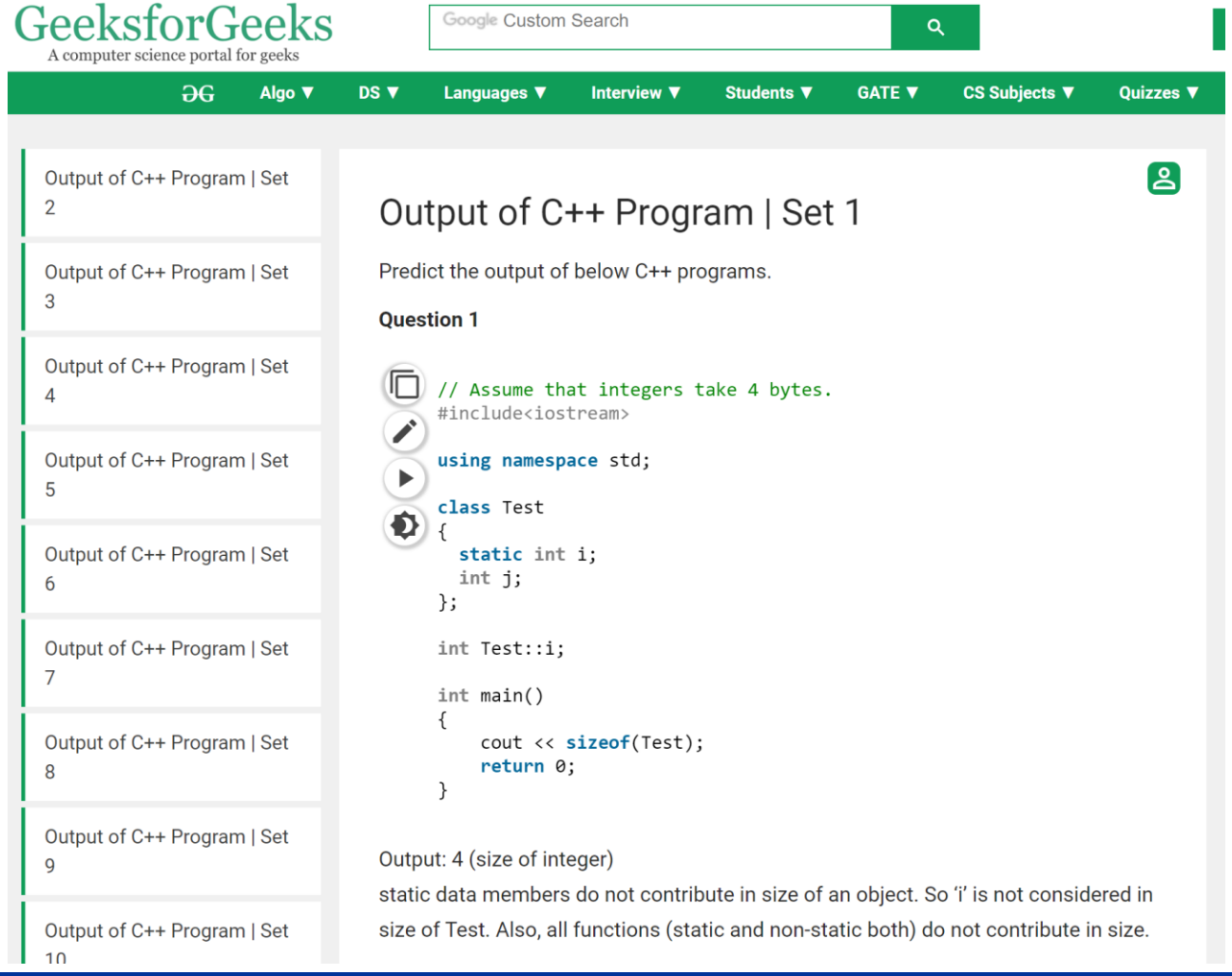
- After you have obtained some senses about C++, finish some questions to impress your understanding!

  1. 150 frequently asked questions, section 3.4 (basic but useful)
  2. C++ interview questions

# Part2: Preparation for different questions

## 3. Output(s) or error(s)s for chunk of codes

- GeeksforGeeks: 24 questions, have a practice!



# Part2: Preparation for different questions

NUS National University of Singapore

## 4. Algorithm question

- Coding Interviews: this book will give you a whole picture of coding interview



- Leetcode: easy and medium are okay for Quant Interviews

# Part2: Preparation for different questions

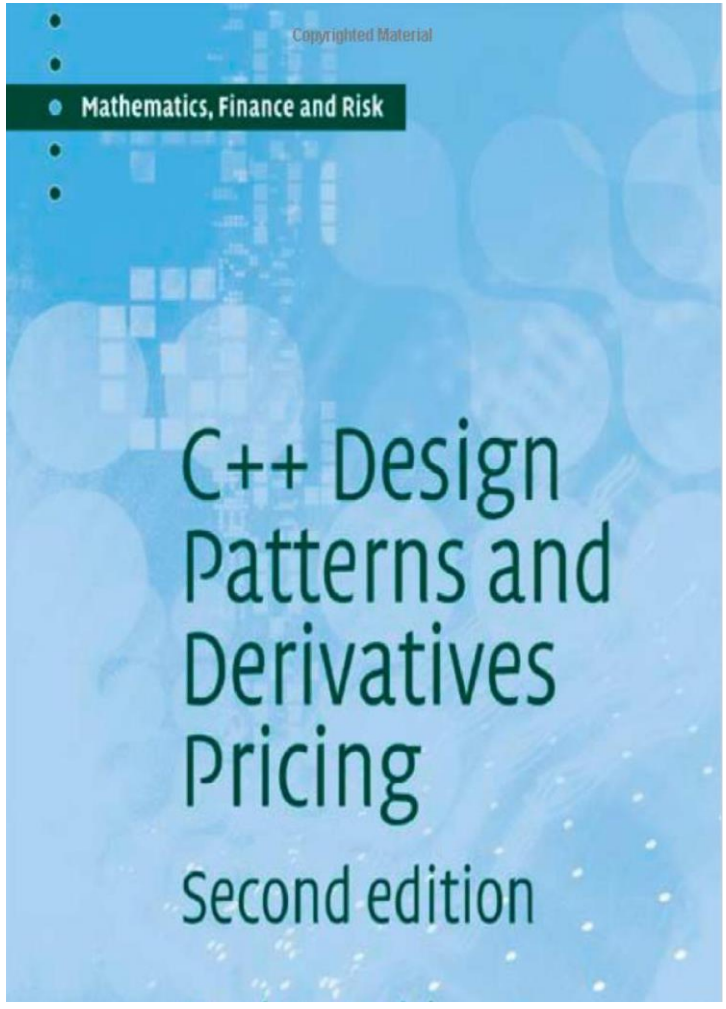
**5. Design pattern of your project (advanced)**

- Slides of QF5204: review again and focus on it's design pattern

- C++ Design Patterns and Derivatives Pricing

The above references will guide you, step by step,

to acquire the rationality of every C++ components



# Part3: Recommended reading

**1. Seamless R and C++ Integration with Rcpp**

- For those who are R user and a beginner of C++, learn how to can integrate R will C++ (low-level language + script language is a standard for banks, hedge funds etc.)

**2. Computer Systems A Programmer's Perspective**

- Machine level understanding of programming languages
- After reading this book, you will know:
  - (i) the components of computer system structure
  - (ii) how OS(operating system deals with your language)
  - (iii) processor architecture
  - (iv) memory hierarchy
  - (v) system-level I/O
  - (vi) network
  - (v) concurrency (process/parallel)


# Thanks for your time!

May you a good luck for final exam