物件導向程式設計

Template (3/3):
Class Template Specialization

Joseph Chuang-Chieh Lin
Dept. CSIE, Tamkang University

Platform

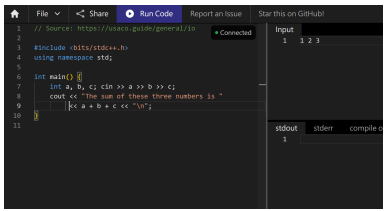
- Dev-C++

  Click here to download.

  Note: Please use this version otherwise you can't compile your programs/projects in Win10.

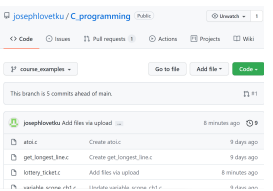
- OnlineGDB (https://www.onlinegdb.com/)

- Real-Time Collaborative Online IDE (https://ide.usaco.guide/)



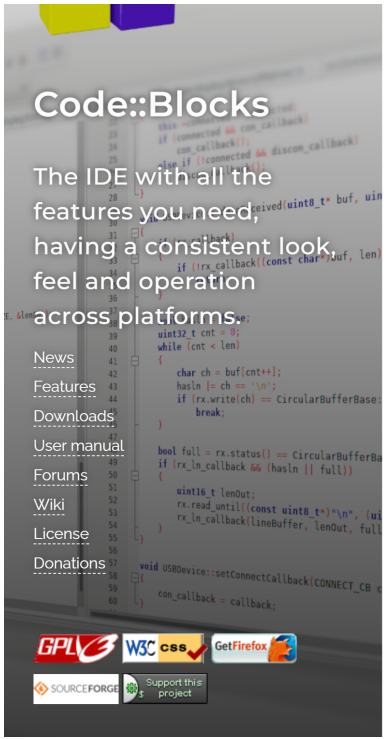
- Other resources:
  - MIT OpenCourseWare - Introduction to C++ [link].
  - Learning C++ Programming [Programiz].
  - GeeksforGeeks [link]

My GitHub page:
click the link here to visit.



Platform/IDE

- https://www.codeblocks.org/



Code::Blocks

Code::Blocks

The free C/C++ and Fortran IDE.

Code::Blocks is a free C/C++ and Fortran IDE built to meet the most demanding needs of its users. It is designed to be very extensible and fully configurable.

Built around a plugin framework, Code::Blocks can be extended with plugins. Any kind of functionality can be added by installing/coding a plugin. For instance, event compiling and debugging functionality is provided by plugins!

If you 're new here, you can read the **user manual** or visit the **Wiki** for documentation. And don't forget to visit and join our **forums** to find help or general discussion about Code::Blocks.

We hope you enjoy using Code::Blocks!

The Code::Blocks Team

Latest news

Migration successful

We are very happy to announce that the process of migrating to the new infrastructure has completed successfully!

Read more

Class Template Specialization

- **Template Specialization**:
  - It is possible to override the template-generated code by providing specific definitions for specific types.

Template Specialization

https://onlinegdb.com/UQn8nDP9o

```
template <class T>
class CHECK {
    public:
        void f() { cout << "CHECK<T>::f()"<< endl ;}
};
```

```
template <>
class CHECK<char> {
    public:
        void f() { cout << "CHECK<char>::f()"<< endl ;}
};
```

```
int main() {
    CHECK<int> c1;
    CHECK<char> c2;

    c1.f();
    c2.f();
    return 0;
}
```

Template Class Partial Specialization

- **Template Partial Specialization**:
  - Generate a specialization of a template class for fewer parameters.

Partial Specialization

https://onlinegdb.com/KPkpfMzQ4

```
template<class T, class U, class V> struct S {
    void foo() {
        cout << "General case" << endl;
    }
};
```

```
template<class U, class V> struct S<int, U, V> {
    void foo() {
        cout << "T = int" << endl;
    }
};
```

```
template<class V> struct S<int, double, V> {
    void foo() {
        cout << "T = int, U = double" << endl;
    }
};
```

```
int main() {
    S<string, int, double> obj1;
    S<int, float, string> obj2;
    S<int, double, string> obj3;

    obj1.foo();
    obj2.foo();
    obj3.foo();
    return 0;
}
```

```
General case
T = int
T = int, U = double
```

Exercise

- Problem 2 of the page here (https://tinyurl.com/2p93tw37).
- **Goal:** Convert a class that is *specialized for integers* into a templated class that can *handle many types*.
- Requirement:
  - Create a templated class named `List` and correctly *initializes*, *manages*, and *de-allocated* an array of *specified length.*
  - Use the **designated** main function:

```
int main() {
    List<int> integers(10);
    for (int i = 0; i < integers.length; i++) {
        integers.set(i, i * 100);
        printf("%d ", integers.get(i));
    }
    printf("\n");
    // this loop should print: 0 100 200 300 400 500 600 700 800 900

    List<Point *> points(5);
    for (int i = 0; i < points.length; i++) {
        Point *p = new Point;
        p->x = i * 10;
        p->y = i * 100;
        points.set(i, p);
        printf("(%d, %d) ", points.get(i)->x, points.get(i)->y);
        delete p;
    }
    printf("\n");
    // this loop should print: (0, 0) (10, 100) (20, 200) (30, 300) (40, 400)
}
```

A typedef struct

```
typedef struct
Point_ {
    int x;
    int y;
} Point;
```

The Output

```
0 100 200 300 400 500 600 700 800 900
(0, 0) (10, 100) (20, 200) (30, 300) (40, 400)
```

Hint

You may refer to the code here as a reference.

```
class IntList {
    int * list;

public:
    int length;
    IntList(int len) {
        list = new int[len];
        length = len;
    }
    ~IntList() {
        delete[] list;
    }
    int get(int index) {
        return list[index];
    }
    void set(int index, int val) {
        list[index] = val;
    }
};
```