قراردادهای برنامه نویسی C++

داخل کلاس را با این ترتیب پر کنید

1. enum declarations
2. struct declarations
3. public functions
4. public slots
5. signal declarations
6. private slots
7. private functions
8. private variables

مثال:

```
class MyClass {
public:
    enum MyEnum {
        MyEnumVal,
        MyEnotherEnumVal,
        ThirdEnumVal
    };

    struct MyStruct {
        int myInt;
        double myDouble;
    };

public: // Public functions
    void publicFunction1() {
        // Function implementation
    }

    void publicFunction2() {
        // Function implementation
    }

public slots:
    // Public slots

signals:
    // Signals

private slots:
    // Private slots

private: // Private functions
```

```
    void privateFunction1() {
        // Function implementation
    }

    void privateFunction2() {
        // Function implementation
    }

private: // Private variables
    int privateInt;
    double privateDouble;
};
```

- به جای چند `if else` اگر میتوانید حتما از `switch` استفاده کنید.
- از قالب `<library.h>` برای `include` کردن کتابخانه های خود زبان و کتابخانه های آماده استفاده کنید و از قالب `"library.h"` برای `include` کردن کتابخانه هایی که در پروژه نوشته شده استفاده کنید.
- برای تعریف `Constructor` یک کلاس اگر میتوانید حتما از List Initializer استفاده کنید.

مثال:

```
...
private:
    int x;
    int y;
public:
    Point(int i, int j):x(i), y(j) {}
    /*  The above use of Initializer list is optional as the
        constructor can also be written as:
        Point(int i = 0, int j = 0) {
            x = i;
            y = j;
        }
    */
...
```

برای members data const non-static of initialization و ref- of initialization members erence حتی کامپایلر ما را مجبور به این کار میکند.

- یک مورد خیلی کاربردی `List Initializer` هم تعیین پارامترهای کلاس پدر یا `Base Class` است. مثال:

```
#include <iostream>
using namespace std;

class A {
    int i;
public:
```

```
    A(int );
};

A::A(int arg) {
    i = arg;
    cout << "A's Constructor called: Value of i: " << i << endl;
}

// Class B is derived from A
class B: A {
public:
    B(int );
};

B::B(int x):A(x) { //Initializer list must be used
    cout << "B's Constructor called";
}

int main() {
    B obj(10);
    return 0;
}
```

استفاده از این روش **Performance** را هم بهبود می دهد!

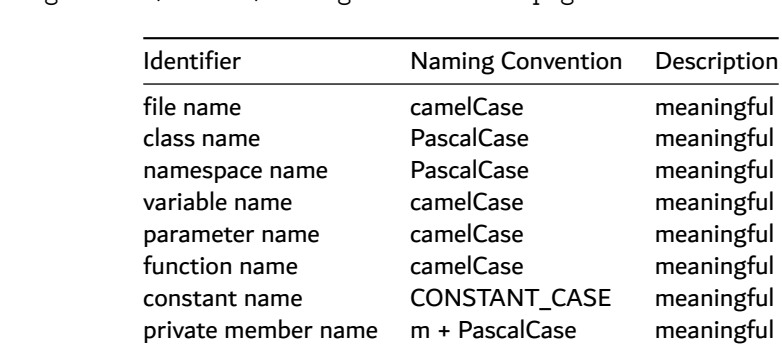
- برای initial کردن متغیرها از {} استفاده کنید. مثال:

```
int x{2};
```

این روش باعث می شود کامپایلر برای کد زیر خطا دهد و این معمولا به ما کمک میکند:

```
unsigned x{-1}; // compiler error
```

قراردادهای نام گذاری



Identifier	Naming Convention	Description
file name	camelCase	meaningful
class name	PascalCase	meaningful
namespace name	PascalCase	meaningful
variable name	camelCase	meaningful
parameter name	camelCase	meaningful
function name	camelCase	meaningful
constant name	CONSTANT_CASE	meaningful
private member name	m + PascalCase	meaningful

Identifier	Naming Convention	Description
typedef name	PascalCase	meaningful
define guard name	CONSTANT_CASE	meaningful
enum member name	PascalCase	meaningful

منابع

https://www.geeksforgeeks.org/when-do-we-use-initializer-list-in-c/

https://google.github.io/styleguide/cppguide.html