OOP for Scientific Computing Notes - SoSe 24

Igor Dimitrov

2024-04-22

Table of contents

Preface

This is a Quarto book.

To learn more about Quarto books visit https://quarto.org/docs/books.

1 Introduction

This is a book created from markdown and executable code.

See Knuth (1984) for additional discussion of literate programming.

Part I

Notes

2 Fundamental Concepts of C++

2.1 Preface

- variables and types
- pointers and references
- control structures
- functions and templates
- classes and inheritance
- namespaces and structure

2.1.1 Variables, Temporaries, Literals

some stuff comes here...

2.1.2 Introducing New Types

- `enum`

```
enum Color = {RED, BLUE, GREEN}
```

- `struct`

2.1.3 Pointers

```
#include <iostream>

int main(int argc, char const *argv[])
{
    int i = 5;
    int *p1 = &i;
    int *p2 = new int;
```

```
    std::cout << "i: " << i << std::endl
              << "*p1: " << *p1 << std::endl
              << "p1: " << p1 << std::endl
              << "&p1: " << &p1 << std::endl
              << "p2: " << p2 << std::endl
              << "*p2: " << *p2 << std::endl;
    delete p2;
    return 0;
}
```

output:

```
i: 5
*p1: 5
p1: 0x7fff8d568184
&p1: 0x7fff8d568188
p2: 0x55c014358eb0
*p2: 0
```

- release memory with `delete`.
- deleting too early -> bugs, too late -> memory leaks

2.1.4 References

References are **aliases for an existing entity**. k

```
#include <iostream>

int main(int argc, const char** argv) {

    int a = 4;
    std::cout << "a: " << a <<std::endl;
    int &b = a;
    b = 5;
    std::cout << "a: " << a << std::endl
              << "b: " << b << std::endl;

    return 0;
}
```

output:

```
a: 4
a: 5
b: 5
```

2.1.5 Rvalue (double) References

Two uses:

- **range-based `for` loops**
- **move semantics**

lvalue references refer to entities, rvalue references refer to literals.

2.1.6 Const-Correctness

Marks something that can't be modified.

```
#include <iostream>

int main(int argc, char const *argv[])
{
    int n = 5;
    const int j = 4;
    const int &k = n; //k can't be modified, equivalently n can't be modified over k
    n++; //but this changes n and indirectly k (because k references n)


    const int *p1 = &n; // modifiable pointer to const int
    int const *p2 = &n; // same thing
    int *const p3 = &n; // constant pointer to modifiable int

    // p1 = &j -- ok
    // *p1 = 3 -- not ok!
    // p3 = &j -- not ok
    // *p3 = 10 -- ok

    std::cout << "n: " << n << std::endl
              << "j: " << j << std::endl
              << "p1: " << p1 << std::endl
              << "p2: " << p2 << std::endl
              << "p3: " << p3 << std::endl;
```

```
    return 0;
}
```

2.1.7 Control Flow

2.1.7.1 If

```
#include <iostream>

int main(int argc, char const *argv[])
{
    int i;
    std::cin >> i;

    if (i % 2 == 0) std::cout << i << " is even" << std::endl;
    else std::cout << i << " is odd" << std::endl;
    return 0;
}
```

2.1.7.2 Switch

```
#include <iostream>

enum Color {RED, BLUE, GREEN};

int main(int argc, char const *argv[])
{
    int i;
    Color c = RED;

    std::cin >> i;

    switch(i) {
        case 0:
            c = RED;
            break;
        case 1 :
```

```
            c = BLUE;
            break;
        case 2 :
            c = GREEN;
            break;
        default :
            std::cout << "error: invalid color" << std::endl;
    }

    std::cout << c << std::endl;

    return 0;
}
```

3 Summary

In summary, this book has no content whatsoever.

References

Knuth, Donald E. 1984. “Literate Programming.” *Comput. J.* 27 (2): 97–111. https://doi.org/10.1093/comjnl/27.2.97.