

Programming in C++

<https://fan1x.github.io/cpp21.html>
tomas.faltin@matfyz.cuni.cz

Programming in C++ - lab 2

<https://fan1x.github.io/cpp21.html>
tomas.faltin@matfyz.cuni.cz

Recap

Homework Example

Class/Struct

- Put all related things (data, functions) together
 - Represents objects in OOP
 - almost everything should belong to a class
- No real difference except for default visibility, inheritance, ...
 - `class` – by default everything `private`
 - `struct` – by default everything `public`
- Internal things → `private`
 - `protected` if need access from a child
- Read-only functions → `const`
 - const-correctness
- Special methods (**constructor**, destructor, ...)

Class Example

```
class calculator {  
    // by default everything is private  
    void sum();  
    void subtract();  
  
public:  
    calculator() { /* default ctor */ }  
    calculator(const std::string &str) () {  
        /* ctor */  
    }  
    void calc(const std::string &str);  
    void print_result() const;
```

```
private:  
    void multiply();
```

can be used
multiple
times

```
protected:  
    void init();
```

```
private:  
};
```

semicolon
at the end!

```
calculator c; // no need for new!  
c.calc("1+2-3");  
c.print_result();
```

```
// calling non-default ctor  
calculator c2("1+2-3");  
c2.print_result();
```

```
// creating a vector  
std::vector<calculator> calcs;
```


Class vs. Struct

- Use class if the class has an invariant; use struct if the data members can vary independently

```
struct coordinate {  
    int x;  
    int y;  
    int z;  
  
    coordinate();  
    coordinate(int x);  
    coordinate(int x, int y);  
    coordinate(int x, int y, int z);  
  
    void set(int x, int y, int z);  
};
```


std::vector<T>

- Beware of time complexity (internally dynamic array)
- vector<bool> optimization

```
#include <vector>
int main() {
    std::vector<int> vi{1, 2, 3, 4, 5, 6}; // [1, 2, 3, 4, 5, 6]
    std::vector<float> vf(5, 0.0f); // [0.0, 0.0, 0.0, 0.0, 0.0]
    std::cout << vi[3] << " " << vf.at(3) << std::endl; // access the 4th element
    std::cout << vi.size();
    vi[3] = 100; vi.at(6) = 600; // access the 4th and 7th element
    vf.push_back(100.0f); vf.emplace_back(200.0f); // insert at the end
    vf.emplace_back(200.0f); // create element at the end
    vf.insert(3, 300.0f); vf.emplace(3, 300.0f); // insert at the specific place
    vf.emplace(3, 300.0f); // create element at the specific place
    vi.pop_back(); // erase the last element
    vf.erase(2); // erase the 3rd element
    vi.clear(); // clear whole container
    vi.reserve(10); // reserve space(=memory) for 10 elements
    vi.resize(10); // actually create 10 elements using default ctor
}
```


3D Matrix for Integers – minimal API

- `ctor()`, `ctor(x, y, z)`
- `set(x, y, z, value)`, `get(x, y, z)`, `print()`
- `set_width()`, `set_length()`, `set_height()`, `get_width()`, `get_length()`, `get_height()`
- `get_matrix(x)`, `get_matrix(y)`, `get_matrix(z)`
- `get_vector(x, y)`, `get_vector(y, z)`, `get_vector(x, z)`
- `clear()` – set all values to 0 (zero)
- `fill_with_value(value)` – set all values to a given value
- `num_zeros()`, `num_negatives()`, `num_positives()`;

3D Matrix for Integers - Hints

- Think about the desing
 - array \rightarrow matrix \rightarrow 3D matrix \rightarrow 4D matrix \rightarrow ... \rightarrow XD matrix
 - Design simple first, then continue to the next level
- No need to focus too much on performance yet
- Focus:
 - Passing arguments: const-references, references, ...
 - const functions
 - class design
 - Decomposition into functions
 - Function reusing
 - private/public

Programming in C++ - lab 1

<https://fan1x.github.io/cpp21.html>
tomas.faltin@matfyz.cuni.cz

Basic information

- Email: tomas.faltin@matfyz.cuni.cz
- Lab's web: <https://fan1x.github.io/cpp21.html>
- ZOOM for distance learning
 - <https://cuni-cz.zoom.us/j/94350923737>
 - Credentials in SIS/mail
- Mattermost
 - Invite link:
https://ulita.ms.mff.cuni.cz/mattermost/signup_user_complete/?id=z1knw5ag6p8nipop1i7iciga6a
 - Use ASAP, might expire eventually
 - Channel: `nprgo41-cpp-english`
- Gitlab
 - <https://gitlab.mff.cuni.cz/>
 - <https://gitlab.mff.cuni.cz/teaching/nprgo41/2021-22/eng>

Communication is the key

- Don't be afraid to ask
 - via email
 - on Mattermost (instant)
 - DM if related to you only
 - Into a channel if others can benefit from it
- If you struggle with something
- If you feel like you might miss a deadline
- Be proactive

Labs credit

- Submitted homeworks before Monday midnight (to Gitlab)
 - Even if not attending!
 - Won't be graded, for a feedback
- Two large homeworks in ReCodex (40 points)
 - Points are included in the final score from the course
 - Smaller HW – 15 points, ~November
 - Larger HW – 25 points, ~December
- Software project
 - Topic must be approved by 28/11/2021
 - First submission: 24/4/2022
 - Final submission: 22/5/2022
 - **All the steps typically mean multiple iterations within multiple days. If you wait for the last minute, there is a chance you won't make it**

Code Requirements

- Consistency
 - Be consistent within the code – keep a single code style
- Cleanness, readability
 - Code doesn't contain commented/dead parts
 - Code should be readable on its own
- Safe, modern
 - E.g., prefer `std::vector<int>` to `new int[]`
- Working
 - OFC, if the code is not working, all the above points are not that important, but they will help you with debugging at least 😊

Why C++

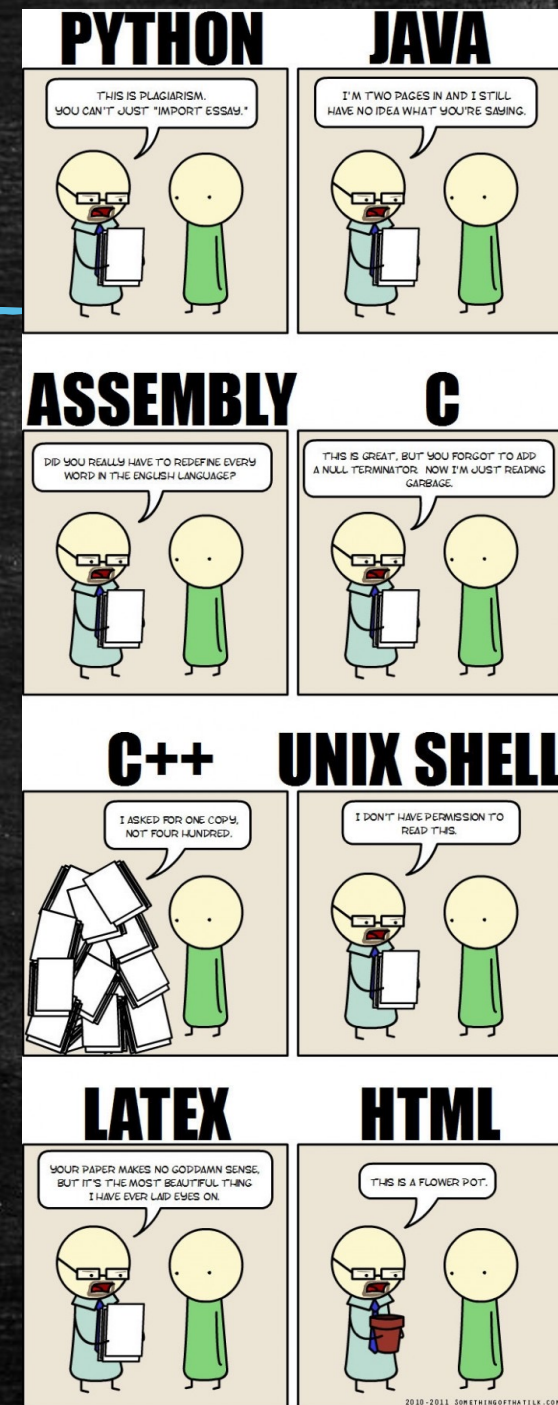
"C makes it easy to shoot yourself in the foot. C++ makes it harder, but when you do, it blows away your whole leg."

-- Bjarne Stroustrup

"It was only supposed to be a joke, I never thought people would take the book seriously. Anyone with half a brain can see that object-oriented programming is counter-intuitive, illogical and inefficient."

-- Stroustrup C++ 'interview' (<https://www-users.cs.york.ac.uk/susan/joke/cpp.htm>)

C++ != speed, C++ ~ control



Working Environment

- Use anything you like 😊
- IDEs
 - Visual Studio
 - License for students at <https://portal.azure.com/...>
 - VS Code
 - Clion
 - Code::Blocks
 - Eclipse
 - ...
- Compilers
 - MSVC, GCC, Clang+LLVM, ICC, ...

C++ (interesting) links

- Reddit, Slack, ...
- <https://en.cppreference.com/w/>
- <http://www.cplusplus.com/>
- <http://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines>
- <https://www.youtube.com/user/CppCon>
- <https://isocpp.org/>
- <http://www.open-std.org/jtc1/sc22/wg21/docs/papers/>
- <https://gcc.godbolt.org/>
- ...

Hello World

```
#include <iostream>
#include <string>

int main() {
    std::string name;
    std::cin >> name;
    std::cout << "Greetings from " << name << std::endl;
    return 0;
}
```


Hello World

Include the libraries
which implements the
used STL constructs
(string, cin, cout)

```
#include <iostream>
#include <string>
```

The main entry
point/function for all
programs. The
execution starts here

```
int main() {
    std::string name;
    std::cin >> name;
    std::cout << "Greetings from " << name << std::endl;
    return 0;
}
```

Read from
standard input
(keyboard)

Write to
standard output
(screen)

All the STL
constructs live
inside `std`
namespace

More Complex Program

```
#include <iostream>
#include <string>
#include <vector>

using namespace std;

int length(const string& s) { ... }

void pretty_print(const vector<string>& a) { ... a[i] ... }

int main(int argc, char** argv) {
    vector<string> arg(argv, argv+argc);
    if (arg.size() > 1 && arg[1] == "--help") {
        cout << "Usage: myprg [OPT]... [FILE]..." << endl;
        return 8;
    }
    pretty_print(arg);
    return 0;
}
```


More Complex Program

```
#include <iostream>
#include <string>
#include <vector>

using namespace std;

int length(const string& s) { ... }

void pretty_print(const vector<string>& a) { ... }

int main(int argc, char** argv) {
    vector<string> arg(argv, argv+argc);
    if (arg.size() > 1 && arg[1] == "--help") {
        cout << "Usage: myprg [OPTION]... [FILE]..." << endl;
        return 8;
    }
    pretty_print(arg);
    return 0;
}
```

Include the whole
std namespace

Passing the
argument by
(const) reference

Arguments of the
program on the
command line

Transform the
arguments into C++
array of strings

Homeworks

1. Hello World
2. A greeting program (use names from arguments)
 - ``hello.exe Adam Eve`` → ``Hello to Adam and Eve``
 - What is inside `args[0]`?
3. Summation of numbers from arguments
 - ``sum.exe 1 2 3 4 5`` → ``15``
 - ``stoi(), stod(), stoX()``
 - Functions for transformation from string to <something>
4. A simple calculator (only for operations + -)
 - ``calc.exe 1+2+3-4`` → ``2``
 - to Gitlab
 - The previous programs are not needed, they should give you a lead