




# Programování v C++ cvičení (2020/21)

[faltin@ksi.mff.cuni.cz](mailto:faltin@ksi.mff.cuni.cz)

<https://fan1x.github.io/cpp20.html>



# Programování v C++ cvičení 3 (14.10.2020)

[faltin@ksi.mff.cuni.cz](mailto:faltin@ksi.mff.cuni.cz)

<https://fan1x.github.io/cpp20.html>



# Operátor „down to“

```
// vypíše 9, 8, 7, ..., 0
void op_downto() {
    int x = 10;
    while (x --> 0) {
        cout << x;
    }
}
```

# class/struct

- Speciální metody: constructor, copy-constructor, move-constructor, destructor, copy-operator, move-operator

```
class C {
    int x = 0;
public:
    C() { cout << "ctor\n"; }
    C(const C &c) : x(c.x) { cout << "copy-ctor\n"; }
    C(C &&c) : x(c.x) { cout << "move-ctor\n"; }
    ~C() { cout << "dtor\n"; }
    C &operator=(const C &c) {
        x = c.x;
        cout << "copy-op\n";
        return *this;
    }
    C &operator=(C &&c) {
        x = c.x;
        cout << "move-op\n";
    }
};
```

# Úkol 1: implementace třídy C

- Implementovat třídu C, aby program vypsal čísla 1, 2, 3, ..., 20
  - NE **exit()** apod...

```
class C { ... }; // implement

// nesahat na věci níže!
void fn_copy(C) {}
void fn_cref(const C&) {}

int main(int argc, char* argv[])
{
    cout << "1";
    C c;
    cout << "5";
    fn_copy(c);
    cout << "10";
    fn_cref(c);
    fn_copy(std::move(c));
    cout << "15";
}
```



# Úkoly

1. Implementovat třídu C
2. Třída pro komplexní čísla
  - naimplementovat všechny speciální metody



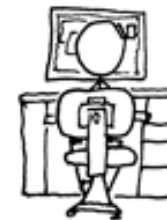
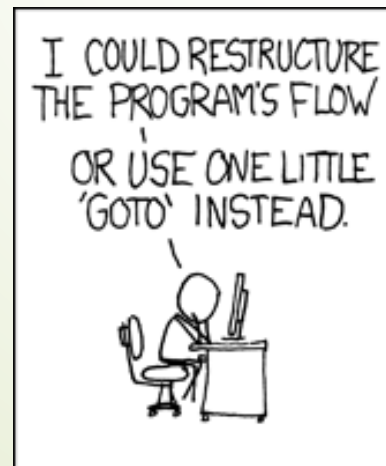
# Programování v C++ cvičení 2 (7.10.2020)

[faltin@ksi.mff.cuni.cz](mailto:faltin@ksi.mff.cuni.cz)

<https://fan1x.github.io/cpp20.html>

# Úkoly (zkušenosti)

- Nepoužívat copy&paste
- Rozdělovat do funkcí
- Místo komentářů používat funkce
- Využívat funkce z STL (např.: `std::stoi`, `std::list<T>`)
  - pozor na složitost funkcí (`std::vector::delete()`)
- Minimalizovat *continue*, *break*, *goto*





# Úkoly (zkušenosti)

- Rozumné pojmenovávání
- Předávání parametrů





# Úkoly (zkušenosti)

- Do GITu pouze zdrojáky, konf. soubory, ...
  - NE: *.obj*, *.log*, *.pdb*, ...
- Další úkoly odevzdat + vytvořit merge request (přidat mě)
  - Návod na stránkách

# Předávání parametrů

- hodnotou (by value): **void fn(string str)**
  - Vytvoří se **kopie**, která se předá do funkce
- odkazem (by reference)
  - reference: **void fn(string &str)**
    - Funkce modifikuje parametr uvnitř
    - Výstupní parametry (pokud nelze návratovou hodnotou)
  - const-reference: **void fn(const string &str)**
    - Předává se parametr, ale nechci vytvářet kopii (pro velké třídy, kde je kopírování drahé)
  - r-value reference: **void fn(string &&str)**
    - Později
- \*ukazatelem (by pointer)
  - **Není to způsob předávání parametrů** (ukazatel je předáván hodnotou)
  - V C-čku - nemá reference

# class/struct (1/2)

- Chceme strukturovat data, funkce „k sobě“

```
class calculator {  
    void sum(); // private by default  
    public:  
        void calc(const std::string &expression); // change internals  
        void print() const; // doesn't change internals  
    private:  
        // ...  
};  
  
calculator c; // no need for `new`  
c.calc("1+2*3/4");
```

- **const** funkce může volat pouze **const** funkci

# class/struct (2/2)

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

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

<http://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#c-classes-and-class-hierarchies>

<http://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#c2-use-class-if-the-class-has-an-invariant-use-struct-if-the-data-members-can-vary-independently>

# std::vector<T>

```
#include <vector>
int main() {
    std::vector<int> vi{1, 2, 3, 4, 5, 6};
    std::vector<float> vf(5, 0.0f);
    std::cout << vi[3] << " " << vf.at(3) << std::endl;
    std::cout << vi.size();
    vi[3] = 100; vi.at(6) = 600;
    vf.push_back(100.0f); vf.emplace_back(200.0f);
    vf.insert(3, 300.0f); vf.emplace(3, 300.0f);
    vi.pop_back();
    vf.erase(3);
    vi.clear();
    vi.resize(10); vi.reserve(100);
}
```

- Pozor na časovou složitost operací
- vector<bool>



# Úkoly

## 1. matice pro čísla

- `set(x, y, value), get(x, y), print()`
  - `set_width(), set_height(), get_width(), get_height()`
  - `get_row(x), get_column(x)` – vrať řádek/sloupec x
  - `get_rows(), get_columns()` – vrátí pole všech řádků/sloupců
  - `clear()` – nastav všechny hodnoty na 0
  - `fill_with_value(value)` – nastav všechny hodnoty na danou hodnotu
  - `reverse()` – prohodí hodnoty z [x, y] na [y, x]
  - `is_negative()` – jsou všechny čísla v matici záporná?
  - `get_negative(), get_positive()` – vrátí všechna negativní/pozitivní čísla v matici
  - `zero_count()` – počet 0 v matici
- 
- POZOR: const metody, předávání parametrů
  - Odevzdat do Gitlabu + merge request



# Programování v C++ cvičení 1 (30.9.2020)

[faltin@ksi.mff.cuni.cz](mailto:faltin@ksi.mff.cuni.cz)

<https://fan1x.github.io/cpp20.html>





# Distanční výuka



- **Web:** <https://fan1x.github.io/cpp20.html>
- **Zoom:** online cvičení
  - Informace k přihlášení v SIS/Nástěnka
- **Slack:** rychlá komunikace se cvičícím/přednášejícím/kolegy
  - Informace k přihlášení v SIS/Nástěnka
- **Gitlab:** odevzdávání úkolů
  - <https://gitlab.mff.cuni.cz/>
- **Recodex:** odevzdávání větších úkolů + automatická oprava
  - <https://recodex.mff.cuni.cz/>



# Požadavky na zápočet



- Dokončené + odladěné příklady ze cvičení **v Gitlabu do pondělí 23:59** před dalším cvičením
  - Ikdyž se neúčastníte cvičení
- 2 DÚ **v ReCodexu**
  - 1. menší úkol: listopad, 15b
  - 2. větší úkol: prosinec, 25b
  - Body se započítávají do zkoušky
- Zápočtový program
  - Téma do **30.11.**
  - 1. odevzdání do **30.4.**
  - Finální odevzdání do **28.5.**
- Invidividuální podmínky je možné domluvit na začátku semestru



# Požadavky na úkoly

- Konzistence (alespoň v rámci jednoho úkolu)
- Čitelný kód
  - Čitelný kód >> komentáře
- Bezpečný kód
  - `std::vector<int> a(20);` >> `int *a = new int[20];`
- Moderní kód
  - `std::array<int, 20> a;` >> `int a[20];`
- Funkčnost



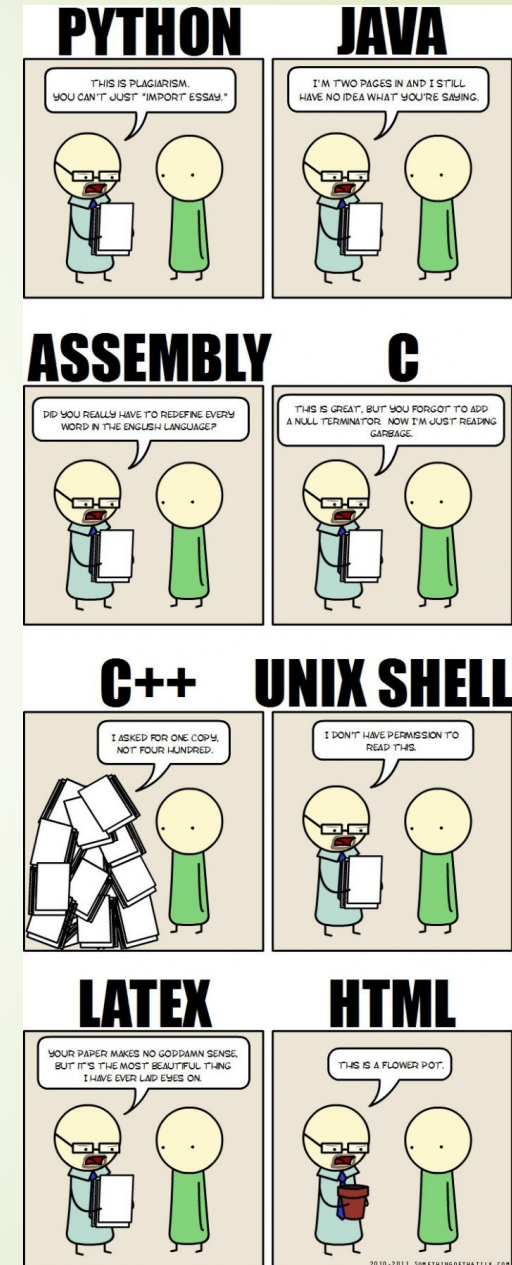
# Bud'te aktivní 😊

- Nebojte se zeptat na Slacku/mailu/...
- Stáže
  - CppCon
  - Google, Microsoft, Oracle, ...
- BP, DP, SWP, PhD

# Proč 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
- C++ is like teenage sex:
  - It's on everyone's mind all the time.
  - Everyone talks about it all the time.
  - Everyone thinks everyone else is doing it.
  - Almost no one is really doing it.
  - The few who are doing it are
    - doing it poorly;
    - sure it will be better next time;
    - not practicing it safely.
- C++ != speed

Source: <http://devhumor.com/media/languages-as-essays>





# Prostředí

## ➤ IDE

- Visual Studio (<https://portal.azure.com/...>)

- Clion

- Code::Blocks

- Eclipse

## ➤ Překladače

- MSVC, GCC, Clang+LLVM, ICC, ...



# C++ (interesting) links



- Reddit, Slack, ...
- <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/>
- <https://en.cppreference.com/w/>
- <http://www.cplusplus.com/>
- ...





# Hello world

```
#include <iostream>
```

```
#include <string>
```

```
int main() {
```

```
    std::string name;
```

```
    std::cin >> name;
```

```
    std::cout << "Greetings from " << name << std::endl;
```

```
    return 0;
```

```
}
```





# Užitečný kód

```
#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;
}
```



# Úkoly 30.9.2020

1. Hello world
2. Program pozdraví všechny lidi (jména zadaná jako argumenty programu)
  - ▀ ``Hello.exe Adam Bedrich Cecilie``
  - ▀ Pozor na první argument, tedy ``arg[0]``
3. Sčítání čísel zadaných jako argumenty
  - ▀ ``std::stoi()`, ...`
4. Jednoduchá kalkulačka nad zadanými argumenty
  - ▀ Jenom čísla a binární operace `+`, `-`, `*`, `/`
  - ▀ ``Calc.exe 1+2*3-4/5``