Písomná skúška z PT: 24. januára 2022

Táto písomná skúška trvá 90 minút. Počas skúšky je povolené používať knihy, poznámky, učebné texty, referencie jazyka C/C++, kompilátor jazyka C/C++. Kódy predpokladajú kompilátor s podporou C++ 17. Kódy boli testované s g++ -std=c++17 (g++ verzia 9.3.0).

Túto písomnú skúšku je potrebné vypracovať samostatne, teda bez pomoci niekoho iného a bez komunikácie s niekým iným. Odhalené podvádzanie pri skúške, napr. kopírovanie odpovedí a riešení (aj ich častí), snaha odovzdať cudzie odpovede a riešenia a pod., môže byť penalizované vylúčením študenta z predmetu, nepridelením a/alebo zrušením bodov, a to aj bodov z cvičení, prípadne aj disciplinárnym konaním v zmysle Študijného poriadku, ktoré môže viesť k vylúčeniu zo štúdia.

1. a) Prečo kód neskompiluje? b) Prečo kód po zakomentovaní presúvacieho (move) konštruktora skompiluje? (6 b.)

```
1  class Token {
2  public:
3    Token() = default;
4    Token(Token&&) = default;
5    ~Token() = default;
6  };
7
8  int main() {
9    Token t0;
10   Token t1 = t0;
11
12   return 0;
13 }
```

---

2. a) Prečo kód neskompiluje? b) Čo je potrebné zmeniť na riadku 9, aby kód skompiloval? Zakomentovanie riadku 9 nie je povolené a zmena musí byť najmenšia možná! (6 b.)

```
1  #include <vector>
2
3  class Token {};
4
5  int main() {
6    std::vector<Token*> v;
7    v.push_back(new Token);
8
9    Token* t = v.begin();
10
11   delete t;
12   return 0;
13 }
```

---

3. a) Prečo je zavolaná len metóda TokenD::foo()? b) Prečo je zavolaný deštruktor triedy TokenB, ale zároveň aj deštruktor triedy TokenD? (8 b.)

```
1  #include <iostream>
2
3  class TokenB {
4  public:
5    virtual void foo() { std::cout << "TokenB::foo()"; }
6    virtual ~TokenB() { std::cout << "~TokenB()"; };
7  };
8
9  class TokenD : public TokenB {
10 public:
11   void foo() override { std::cout << "TokenD::foo()"; }
12   ~TokenD() override { std::cout << "~TokenD()"; };
13 };
14
15 int main() {
16   TokenB* t0 = static_cast<TokenB*>(new TokenD);
17   t0->foo();
18
19   delete t0;
20   return 0;
21 }
```

---

4. a) Prečo je deštruktor zavolaný 3-krát? b) Na ktorom riadku vznikne objekt typu Token, ktorý zanikne ako prvý? c) Prečo tento objekt zanikne pred ostatnými dvoma objektmi? d) Prečo je prednastavený (default) konštruktor zavolaný len 1-krát? (8 b.)

```
1  #include <memory>
2  #include <iostream>
3
4  class Token {
5  public:
6    Token() { std::cout << "Token()"; }
7    ~Token() { std::cout << "~Token()"; }
8  };
9
10 template <class T>
11 std::unique_ptr<T> foo(T t0) {
12   return std::unique_ptr<T>(new T(t0));
13 };
14
15 int main() {
16   Token&& t = Token();
17
18   auto ptr = foo<Token>(t);
19
20   return 0;
21 }
```

---

5. a) Prečo tento kód neskompiluje? b) Čo je v kóde potrebné zmeniť a/alebo doplniť, aby kód skompiloval? Aké riešenie je najefektívnejšie? c) Prečo je pri použití std::vector táto zmena potrebná? (8 b.)

```
1  #include <vector>
2  #include <memory>
3  #include <iostream>
4
5  class Token {
6  private:
7    std::unique_ptr<std::vector<int> > ptr = std::make_unique<std::vector<int> >();
8  public:
9    Token() = default;
10   Token(Token& t0) noexcept { std::cout << "Token(Token&)"; }
11 };
12
13 int main() {
14   std::vector<Token> v;
15
16   for(int i = 0; i<10; ++i) {
17     v.emplace_back();
18   }
19
20   return 0;
21 }
```

---

6. Časová zložitosť metódy find na riadku 11 je O(log n), kde n je počet prvkov v kontaineri. a) Akú dátovú štruktúru a kontajner z STL knižnice musíme použiť, aby sme prvok v kontaineri mohli nájsť v konštantnom čase (v priemere)? b) Aké výhody a nevýhody má takáto dátová štruktúra? (8 b.)

```
1  #include <set>
2  #include <iostream>
3
4  int main() {
5    std::set<int> s;
6
7    for(int i=0; i<1000; ++i) {
8      s.insert(i);
9    }
10
11   auto it = s.find(100);
12
13   return 0;
14 }
```

---

7. Prečo riadok 15 skompiluje, a riadok 17 neskompiluje? (8 b.)

```
1  #include <iostream>
2
3  class Token {
4  private:
5    int a{1};
6  public:
7    std::ostream& operator<<(std::ostream& os) {
8      os << a;
9      return os;
10   }
11 };
12
13 int main() {
14   Token t;
15   t << std::cout;
16
17   std::cout << t;
18
19   return 0;
20 }
```

---

8. a) Prečo tento kód neskompiluje? b) Ako je potrebné zmeniť definíciu členov triedy Token tak, aby kód skompiloval, a zároveň aby objekty boli v std::set s uložené podľa použitého komparátora? (8 b.)

```
1  #include <iostream>
2  #include <set>
3
4  class Token {
5  public:
6    int a();
7    Token(int i0) : a(i0) {};
8    bool operator<(const Token& t0) { return a < t0.a; }
9  };
10
11 int main() {
12   bool (*comp)(const Token&, const Token&) = [](const Token& t0, const Token& t1) -> bool {
13                                                 return t0 < t1;
14                                               };
15
16   std::set<Token, decltype(comp)> s(comp);
17
18   for(int i=0; i<3; ++i) { s.emplace(i); }
19
20   for(auto i : s) { std::cout << i.a << std::endl; }
21
22   return 0;
23 }
```
