Písomná skúška z PT: 8. januára 2021

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. Túto písomnú skúšku je potrebné vypracovať samostatne.

1. Deštruktor triedy TokenB je virtuálny, napriek tomu je deštruktor triedy TokenD zavolaný len raz. Prečo? (8 b.)

```
#include <iostream>

class TokenB {
public:
  virtual ~TokenB() { std::cout << "~TokenB()"; }
};

class TokenD : public TokenB {
public:
   ~TokenD() { std::cout << "~TokenD()"; }
};


int main() {
  TokenD d;
  TokenB b = static_cast<TokenB>(d);

  return 0;
}
```

---

2. Prečo t0 + t1 kompilátor skompiluje a t0 + t1 + t2 už kompilátor neskompiluje? (8 b.)

```
#include <iostream>

class Token {
public:
  int a{1};
  int operator+(const Token& t) {
    return this->a + t.a;
  }
};

int main() {
  Token t0, t1, t2;

  std::cout << t0 + t1;

  std::cout << t0 + t1 + t2;

  return 0;
}
```

---

3. Prečo kód neskompiluje? Čo je potrebné pridať, aby tento kód skompiloval? Pridanie sa v tomto prípade rozumie nový kód, a nie modifikácia už existujúceho. (4 b.)

```
#include <iostream>

class Token {
private:
  int a{1};
public:
  Token& operator++() {
    ++a;

    return *this;
  }
};

int main() {
  Token t0;
  Token t1 = ++t0;

  std::cout << t1;

  return 0;
}
```

---

4. Prečo sa nezavolá kopírovací a ani presúvací konštruktor? (8 b.)

```
#include <iostream>

class Token {
public:
  Token() = default;

  Token(const Token& t) {
    std::cout << "Token(const␣Token&␣t)";
  }

  Token(Token&& t) {
    std::cout << "Token(Token&&␣t)";
  }
};

int main() {
  Token* t0 = new Token;

  Token* t1 = t0;

  return 0;
}
```

---

5. Prečo tento kód neskompiluje? (8 b.)

```
#include <vector>
#include <memory>

class Token {
public:
  std::unique_ptr<std::vector<int> > ptr;
  Token() : ptr(std::make_unique<std::vector<int> >()) {}
};

int main() {
  Token t0;
  Token t1 = t0;

  return 0;
}
```

---

6. Prečo je zavolaný len kopírovací konštruktor? Čo je potrebné zmeniť, aby pri zmene veľkosti vektora vec bol zavolaný presúvací konštruktor (kopírovací konštruktor nesmie byť zakomentovaný a ani označený ako delete)? (8 b.)

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

class Token {
public:
  Token() = default;
  Token(const Token& t) { std::cout << "Token(const␣Token&␣t)"; }
  Token(Token&& t) { std::cout << "Token(Token&&␣t)"; }
};

int main() {
  std::vector<Token> vec;

  vec.emplace_back();
  vec.emplace_back();
  vec.emplace_back();

  return 0;
}
```

---

7. Prečo tento kód neskompiluje? Aký konštruktor by ste pridali v triede TokenD, aby tento kód skompiloval? (8 b.)

```
1  class TokenB {};
2  class TokenD : public TokenB {};
3
4  int main() {
5    TokenB b;
6    TokenD d = static_cast<TokenD>(b);
7
8    return 0;
9  }
```

---

8. Prečo v tomto prípade nedokážeme zachytiť výnimku? Prečo kód vždy predčasne skončí? (8 b.)

```
1  #include <iostream>
2
3  class Token{
4  public:
5    ~Token() {
6      throw 1;
7    }
8  };
9
10 int main() {
11   Token* t = new Token;
12
13   try{
14     delete t;
15   } catch(int a) {
16     std::cout << "Exception!";
17   }
18
19   return 0;
20 }
```
