Písomná skúška PT: 18. decembra 2024

Vypni zvonenie na mobilnom telefóne!

Meno a priezvisko: ⟸

Písomná skúška trvá 90 minút. Kódy na písomnej skúške budú testované s g++ -std=c++20 (g++ verzia 13.1.0).

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, prípadne aj disciplinárnym konaním v zmysle Študijného poriadku, ktoré môže viesť k vylúčeniu zo štúdia.

1. Aké je poradie volania konštruktorov? Odôvodnite odpoveď. (6 b.)

Písomné odôvodnenie riešenia (max. 30 slov):

```
1  #include <iostream>
2
3  class TokenB {
4  public:
5    TokenB() {
6      std::cout << "TokenB()";
7    }
8    TokenB(const TokenB&) {
9      std::cout << "TokenB(const␣TokenB&)";
10   }
11 };
12
13 class TokenD : public TokenB {
14 public:
15   TokenD() {
16     std::cout << "TokenD()";
17   }
18   TokenD(const TokenD&) {
19     std::cout << "TokenD(const␣TokenD&)";
20   }
21 };
22
23 int main() {
24   TokenB t0 = TokenD();
25
26   return 0;
27 }
```

---

2. (a) Aký je rozdiel medzi kódom na riadkoch 8-10? (b) Koľko nových objektov vznikne na riadkoch 8-10? Ako vzniknú tieto nové objekty? (6 b.)

Písomné odôvodnenie riešenia (max. 30 slov):

```
1  class TokenB {};
2
3  class TokenD : public TokenB {};
4
5  int main() {
6    TokenD t;
7
8    TokenB t0 = static_cast<TokenB>(t);
9    TokenB t1 = (TokenB) t;
10   TokenB t2 = TokenB(t);
11
12   return 0;
13 }
```

3. (a) Čo je potrebné pridať, aby pri zmene veľkosti vektora v (resize) nastalo efektívne presunutie objektov typu Token? (b) Prečo je vektor v0 potrebné efektívne presunúť? Vysvetlite implementáciu presunutia vektora v0 (napíšte kód presunutia vektora v0 bez použitia =default). (8 b.)

Písomné odôvodnenie riešenia (max. 30 slov):

```
#include <vector>

class Token {
private:
  std::vector<int> v0 = {1,2,3};
public:
  Token() = default;
  Token(const Token&) = default;
};

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

  for(int i=0; i<10; ++i) {
    v.emplace_back();
  }

  return 0;
}
```

---

4. (a) Aký je rozdiel medzi metódami foo v TokenA a TokenB? (b) Aký následok má použitie kľúčového slova override? (c) Prečo používame override? Ak kľúčové slovo override nepoužijeme, aké následky to môže mať? (8 b.)

Písomné odôvodnenie riešenia (max. 30 slov):

```
class Token {
public:
  virtual void foo() {}
};

class TokenA : public Token {
public:
  void foo() {}
};

class TokenB : public Token {
public:
  void foo() override {}
};
```

5. (a) Aký je rozdiel medzi push_front a emplace_front? (b) Prečo push_front neskompiluje, ale emplace_front skompiluje? (c) Prečo emplace_front nevyžaduje žiaden vstupný parameter? (8 b.)

Písomné odôvodnenie riešenia (max. 30 slov):

```
#include <deque>

class Token {
public:
  Token() = default;
  Token(const Token&) = delete;
  Token& operator=(const Token&) = delete;
};

int main() {
  std::deque<Token> dq;
  Token t;

  dq.push_front(t);
  dq.emplace_front();

  return 0;
}
```

---

6. (a) Na inštancii triedy Container zavolajte metódu foo (napíšte potrebný kód). (b) Vysvetlite, aké zmeny sú potrebné, aby takéto zavolanie bolo možné. (c) Vysvetlite prečo sú tieto zmeny potrebné. (d) Prečo potrebujeme typový parameter T? Môžeme použiť iné označenie, napr. D namiesto T? (8 b.)

Písomné odôvodnenie riešenia (max. 30 slov):

```
#include <deque>

template <typename T>
class Container {
private:
  T value;
public:
  Container() = default;
  Container(T v) : value(v) {}
  void foo() {}
};

int main() {
  Container<std::deque<int> > container();

  return 0;
}
```

7. (a) Pre triedy TokenB, TokenD0 a TokenD1 doplňte kód, ktorý umožní neskoré viazanie (late binding, runtime polymorphism). Vysvetlite riešenie. (b) Prečo potrebujeme neskoré viazanie? (c) Aký je rozdiel medzi skorým a neskorým viazaním? (8 b.)

Písomné odôvodnenie riešenia (max. 30 slov):

```
1  class TokenB {
2  public:
3    void foo() {}
4  };
5
6  class TokenD0 : public TokenB {
7  public:
8    void foo() {}
9  };
10
11 class TokenD1 : public TokenB {
12 public:
13   void foo() {}
14 };
15
16 int main() {
17   TokenD0* t0 = new TokenD0;
18   TokenD1* t1 = new TokenD1;
19
20   TokenB* b0 = t0;
21   TokenB* b1 = t1;
22
23   b0->foo();
24   b1->foo();
25
26   delete t0;
27   delete t1;
28
29   return 0;
30 }
```

---

8. (a) Aký je rozdiel medzi std::vector and std::deque? Aký je rozdiel v ich vnútornej implementácii? (b) Kedy je vhodnejšie použiť std::vector, a kedy je vhodnejšie použiť std::deque? (c) Akú časovú zložitosť má indexový prístup (random access) pre std::vector a std::deque? (d) Aký druh kontainera sú std::vector a std::deque? (8 b.)

Písomné odôvodnenie riešenia (max. 50 slov):