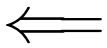
Písomná skúška PT: 19. decembra 2022

Vypni zvonenie na mobilnom telefóne!

Meno a priezvisko, ID:



Písomná skúška trvá 90 minút. Počas skúšky je povolené používať poznámky v rozsahu 5 listov vo formáte A4, ktoré môžete využiť obojstranne, môžete mať na nich ručne písané poznámky, alebo vytlačený text (aj C++ kód) s ľubovolnou veľkosťou písma. Počas skúšky nie povolené používať kompilátory, počítače, iné elektronické zariadenia. Okrem poznámok v povolenom rozsahu nie je povolené používať knihy, učebné texty, referencie jazyka C++. Kódy na písomnej skúške budú testované s g++ -std=c++17 (g++ verzia 11.3.0).

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

1. Ak zmeníme štandardnú definíciu std::set (pozri dole), aké dôsledky to bude mať pre používanie tohto asociatívneho kontajnera? (6 b.)

Štandardná definícia:

Písomné odôvodnenie riešenia (max. 20 slov):

```
template<
    class Key,
    class Compare = std::less<Key>,
    class Allocator = std::allocator<Key>
> class set;
```

Zmenená definícia:

```
template<
    class Key,
    class Compare,
    class Allocator
> class set;
```

---

2. Čo je potrebné doplniť, aby tento kód skompiloval? (6 b.)

Písomné odôvodnenie riešenia (max. 20 slov):

```
#include <iostream>

class Master {
public:
  Master(const Master& m) {
    std::cout << "Master(const␣Master&)";
  }
  class Helper {
  public:
    Helper(const Helper& h) {
      std::cout << "Helper(const␣Helper&)";
    }
  };
};

int main() {
  Master::Helper h0;

  return 0;
}
```

3. Kód neskompiluje na riadku 6. Ak by konštruktor Token(Token) bolo povolené skompilovať, v kóde by nastalo rekurzívne volanie tohto konštruktora. Prečo by nastalo rekurzívne volanie tohto konštruktora? (8 b.)

```
1  #include <iostream>
2
3  class Token {
4  public:
5    Token() = default;
6    Token(Token t) { //error
7      std::cout << "Token(Token)";
8    }
9  };
10
11 int main() {
12   Token t0;
13   Token t1 = t0;
14
15   return 0;
16 }
```

Písomné odôvodnenie riešenia (max. 20 slov):

---

4. Kód nie je skompilovateľný. Prečo? (8 b.)

```
1  #include <set>
2  #include <memory>
3
4  class Token {
5  public:
6    int a{1};
7  };
8
9  int main() {
10   std::set<std::unique_ptr<Token> > s;
11   std::unique_ptr<Token> ptr = std::make_unique<Token>();
12
13   s.insert(ptr); //error
14
15   return 0;
16 }
```

Písomné odôvodnenie riešenia (max. 20 slov):

5. Čo je potrebné doplniť v definícii triedy TokenB tak, aby bol možný dynamic_cast na riadku 15? Prečo je takéto doplnenie potrebné? (8 b.)

```
1  class TokenB {
2  public:
3    int a{1};
4  };
5
6  class TokenD : public TokenB {
7  public:
8    int b{1};
9  };
10
11 int main() {
12   TokenD* td = new TokenD;
13   TokenB* tb = dynamic_cast<TokenB*>(td); //OK
14
15   TokenD* td0 = dynamic_cast<TokenD*>(tb); //error
16
17   delete td;
18   return 0;
19 }
```

Písomné odôvodnenie riešenia (max. 20 slov):

---

6. Koľkokrát bude zavolaný kopírovací konštruktor? Koľkokrát bude zavolaný deštruktor? Odôvodnite odpoveď. (8 b.)

```
1  #include <iostream>
2  #include <vector>
3
4  class Token {
5  public:
6    Token() = default;
7    Token(const Token& t) {
8      std::cout << "Token(const␣Token&)";
9    }
10   ~Token() {
11     std::cout << "~Token()";
12   }
13 };
14
15 int main() {
16   std::vector<Token> v;
17
18   v.emplace_back();
19
20   return 0;
21 }
```

Písomné odôvodnenie riešenia (max. 20 slov):

7. Prečo potrebujeme lambda funkciu hash? Prečo potrebujeme lambda funkciu equal? Ako je potrebné zmeniť hash, aby nastalo hašovanie do k košov, pričom k je prvočíslo (napr. 11)? Prečo je potrebné použiť decltype(hash) a decltype(equal)? (8 b.)

```
#include <unordered_set>

class Token {
public:
  int a{1};
};

int main() {
  auto hash = [](const Token& a) -> std::size_t {
    return 1;
  };

  auto equal = [](const Token& a, const Token& b) -> bool {
    return a.a == b.a;
  };

  std::unordered_set<Token, decltype(hash), decltype(equal)> s(2, hash, equal); //2=počiatočný počet košov
  Token t0;

  s.emplace(t0);

  return 0;
}
```

Písomné odôvodnenie riešenia (max. 20 slov):

---

8. a) Prečo je potrebné, aby funkcia plusOne bola const? Čo by nastalo, ak by táto funkcia nebola const? b) Čo by nastalo, ak by funkcia foo nemala vstupný parameter const Token&, ale Token&? (8 b.)

```
#include <iostream>

class Token {
private:
  int a{1};
public:
  int plusOne() const {
    return this->a + 1;
  }
};

int foo(const Token& t) {
  return t.plusOne();
}

int main() {
  Token t0;
  std::cout << foo(t0) << foo(Token{});

  return 0;
}
```

Písomné odôvodnenie riešenia (max. 20 slov):