

VYBRANÉ FUNKCIONALITY STANDARDU C++14

Studijní materiály

OBSAH

1	ÚVOD	3
2	POPIS VYBRANÝCH ZMĚN	4
2.1	FUNKCIONALITY SPOJENÉ S <i>AUTO</i>	4
2.1.1	Return type deduction	4
2.1.2	<code>decltype(auto)</code>	4
2.2	LAMBDA FUNKCE	5
2.2.1	Inicializace lambda captures	5
2.2.2	Generická lambda.....	5
2.3	ŠABLONY PROMĚNNÝCH	5
2.4	PŘEHLEDNĚJŠÍ ZÁPISY PROMĚNNÝCH	5
2.4.1	Znak oddělení číselného řádu.....	5
2.4.2	Binární literály	5
2.5	AGREGÁTNÍ INICIALIZACE	6
3	ZADÁNÍ SAMOSTATNÉ PRÁCE	7

1 ÚVOD

C++14 je malým standardem, který přinesl pouze menší úpravy, bug fixy a drobná vylepšení funkcí představených ve standardu C++11. Jedny z hlavnějších přínosů jsou dedukce návratového typu, funkce `decltype(auto)`, generické lambda funkce a inicializace lambda captures.

2 POPIS VYBRANÝCH ZMĚN

2.1 Funkcionality spojené s *auto*

2.1.1 Return type deduction

C++14 přidává pár nových funkcionalit k typu *auto* představenému v C++11. Nově nemusíme u funkcí specifikovat koncový návratový typ. Dedukce funguje pro funkce i lambdy.

// C++11

```
auto func1(int const i) -> int
```

```
{ return 2*i; }
```

// C++14

```
auto func2(int const i)
```

```
{ return 2*i; }
```

2.1.2 `decltype(auto)`

V C++14 lze nově „zkombinovat“ typové dedukce *auto* a `decltype`. Pokud použijeme `decltype(auto)`, odvozený typ bude obsahovat i typové modifikátory, jako např. `const`. *Auto* určuje typ který se má odvodit, zatímco `decltype` určuje že pro odvození typu se mají používat `decltype` pravidla. Stejně jako `decltype`, i `decltype(auto)` je užitečný hlavně při generickém programování.

```
const int a = 0;
```

```
auto b = a; // b je typu int
```

```
decltype(auto) c = a; // c je typu const int
```

2.2 Lambda funkce

2.2.1 Inicializace lambda captures

Lambda captures (proměnné mimo lambda) mohou být nově inicializované. Inicializace se provede když je lambda vytvořena, ne zavolána. Inicializace je užitečná např. pokud chceme v lambda funkci použít členy třídy, jelikož je v *captures* můžeme inicializovat samy sebou.

```
auto lambda = [value = 1] {return value;};
```

2.2.2 Generická lambda

Od C++14 mohou parametry lambda funkce být typu *auto*:

```
auto lambdaFunc = [](auto x) { return x; };
```

2.3 Šablony proměnných

V C++14 mohou být vytvořeny šablony na proměnné:

```
template<class T>  
constexpr T pi = T(3.1415926535897932385);
```

2.4 Přehlednější zápisy proměnných

2.4.1 Znak oddělení číselného řádu

Pro lepší čitelnost můžeme při zápisu číselné proměnné hodnotu rozdělit znakem ‘.

```
int a = 1'000'000;
```

2.4.2 Binární literály

Integery mohou nově nabývat formy čísla o základu 2. Zápis čísla musí začínat znaky 0b nebo 0B.

```
int x = 0b1010      // = 10
```

2.5 Agregátní inicializace

Následující struktura není v C++11 agregace, protože defaultně inicializujeme proměnné. Od C++14 se již o agregaci jedná, a můžeme využít inicializace se složenými závorkami.

```
struct S {  
    int x = 1;  
    int y = 2;  
};  
S s1{11,12};
```

3 ZADÁNÍ SAMOSTATNÉ PRÁCE

Upravte vaše řešení úkolu 1 pro standard C++11 tak, aby:

- Funkce vypisující údaje o studentovi místo variadické šablony využívala generickou lambda funkci.
- Funkce pro výpočet průměru známek využívala inicializace lambda captures