Razvoj softvera

Vežbe 5

Lambda funkcije

- Lambda funkcije – anonimna funkcija (anonimni funkcionalni objekat)
- **[*glava*] (*argumenti*) {*telo_funkcije*}**
- **[*glava*] (*argumenti*) -> *tip_povratne_vrednosti* {*telo_funkcije*}**

```
[](x arg1,  x arg2) {return izraz;}
```

Lambda funkcije

- Da bi promenljiva x bila vidljiva unutar lambde, mora biti precizirana unutar glave. Može biti uhvaćena
  - po vrednosti [x]
  - po referenci [&x]
- Mogu se uhvatiti sve korišćene promenljive
  - po vrednosti [=]
  - ili referenci [&]

Algotirmi

- template <class BidirectionalIterator, class UnaryPredicate> BidirectionalIterator **stable_partition** (BidirectionalIterator first, BidirectionalIterator last, UnaryPredicate pred);

Algotirmi

- Sortira elemente u opsegu [first, last) tako da svi elementi za koje predikat pred vraća vrednost *true* prethode elementima za koje predikat pred vraća vrednost. Poredak elemenata u okviru svake grupe se zadržava.
- Povratna vrednost: iterator na prvi element druge grupe ili *last* ukoliko je druga grupa prazna
- Zaglavlje <algorithm>

Iteratori

- begin(arg) – iterator na prvi element u sekvenci
- end(arg) – iterator na element posle-poslednjeg u sekvenci

Algoritmi

- template <class ForwardIterator, class UnaryPredicate> ForwardIterator **remove_if** (ForwardIterator first, ForwardIterator last, UnaryPredicate pred);
- Transformiše opseg [first,last) u ospeg za koji važi da su svi elementi za koje predikat *pred* vraća vrednost *true* uklonjeni. Vraća iterator na novi kraj tog opsega.
- Zaglavlje <algorithm>

Lvalue i rvalue

- Lvalue – izraz koji može da bude na levoj ili na desnoj strani izraza
  - Izraz koji referiše na lokaciju u memoriji i čija se adresa može dobiti korišćenjem operatora &
- Rvalue - izraz koji može da bude samo na desnoj strani izraza
  - Rezultat daje privremeni objekat

Lvalue i rvalue

```
int a=42;
int b=43;
```

```
//a i b su lvalue
a=b;
b=a;
a=a*b;
int* p=&a;
int* p1 = &(a+1); ???
```

```
//a*b je rvalue
int c =a*b;
a*b=42; ???
```

Lvalue i rvalue

```
int x;
int& getRef ()
{
        return x;
}
getRef() = 4;
//vraca referencu na
   globalu promenljivu
```

```
int x;
int getVal ()
{
    return x;
}
getVal();
//vraca rvalue
```

Lvalue i rvalue

```
string getName ()
{
   return "Alex";
}

string name = getName();  //ok
const string& name = getName(); // ok
string& name = getName(); // nije ok
```

Rvalue referenca

- Rvalue reference je referenca koja će se vezati samo za privremene objekte
  - Koristi se za detektovanje da li je vrednost privremeni objekat ili ne
  - Koriste sintaksu && umesto &
  - const string&& name = getName();
    string&& name = getName();

Lvalue i rvalue

- Pisanje funkcija koje kao argument primaju lvalue ili rvalue reference

```
printReference (const String& str)
{
	cout << str;
}
printReference (String&& str)
{
	cout << str;
}
```

Lvalue i rvalue

```
string me( "alex" );
printReference(  me ); // poziva prvu f-ju

printReference( getName() );  //poziva drugu f-ju
```

Lvalue i rvalue

- **forward**(arg) – vraća rvalue referencu na arg, ako argument nije lvalue referenca, a ako jeste onda vraća arg.

- **std::bind –** vezuje argumente funkcije
- template <class Fn, class... Args> **bind** (Fn&& fn, Args&&... args);
- Može se koristiti za parcijalnu aplikaciju, tj. za definisanje nekih argumenata dok ostale ostavljamo za kasnije

- Prvi argument je funkcija kojoj želimo neke argumente da "vežemo", posle toga definisemo ili "rupe " ili konkretne vrednosti
- **std::placeholders** prostor imena u kome su deklarisani objekti _1, _2, ... koji se koriste za definisanje "rupa“ u **std::bind**
  - _1 se zamenjuje prvim argumentom u pozivu funkcije, _2 drugim, _3 trećim ...