Употреба на итератори – час 1

Итераторите се слични на покажувачите и може да се зголемуваат со ++, дереференцитаат со * и дасе споредуваат со користење на !=.

Вектори и итератори

Контејнер е секоја податочна структура којашто содржи елементи од ист тип. Листата, векторот, стекот и редот се контејнери.
До елементите во рамки на контејнерите се пристапуваат преку итератори. Така на пример за да декларираме итератор за вектор со целобројни вредности се користи кодот

vecor<int>::iterator

Декларација на итератор за контејнер од тип C:
C::iterator p
- Поместување на итераторот кон следниот елемент од контејнерот се врши со
p++
- Самиот итератор p е покажувач. Вредноста кон која покажува p е
***p**
- За поставување на итераторот да покажува кон првиот елемент од контејнерот C се користи методата
c.begin()
- За поставување на итераторот да покажува кон последниот елемент од контејнерот C се користи методата
c.end()

Пр.

```
1.  #include <iostream>
2.  #include <vector>
3.  using namespace std;

4.  int main()
5.  {   vector <int> v;
6.  for(unsigned i=0;i<15;i++)
7.  v.push_back(i);
8.  vector<int>::iterator it;
9.  for(it=v.begin(); it!=v.end();it++)
10. cout << *it << ' ';
11. cout << endl;
12. return 0;
13. }
```

Резултатот од извршувањето на програмата е:
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

Во линијата 9 од програмата низ векторот v наместо со бројач, се изминува со итераторот it. Декларацијата на итераторот it е:

vector<int>::iterator it;

каде **iterator** е името на типот, а нотацијата :: се користи да се опредиели на која класа и припаѓа. Во линијата 9 од програмата се иницијализира **iterator** на почетокот од векторот со **it=v.begin();** Функцијата **begin();** враќа итератор на првиот елемент од векторот, а функцијата **end()** враќа итератор на последниот елемент. Со помош на операторот ++ се овозможува итераторот да покажува на следниот елемент. Со помош на операторот != се проверува дали it е различен од некој итаратор.

Во линијата 10 се печатат елементите на векторот со помош на итератор и со користење на операторот * за пристап до елементот на кој покажува итераторот.

Разликата меѓу итератори и покажувачи во C++:

- Покажувачот е променлива која ја содржи адресата на друга променлива, т.е. мемориската локација на другата променлива. Како и секоја друга променлива или константа, покажувачот мора да се декларира пред да се користи.
- Итератор е било кој објект кој покажувајќи на некој елемент од опсегот на елементи (може да е поле или контејнер), ја има способноста да ги измине сите елементи од опсегот.

```
#include <iostream>
#include <vector>
using namespace std;
int main()
{
   int n,p,l;
   // deklariranje na vektor
   vector<int> v;
      cout<<"vnesi broj na elementi"<<endl;
       cin>>n;
       for(int p=0;p<n;p++)
       {
          cout<<"vnesi element"<<endl;
          cin>>l;
          v.push_back(l);
       }
   // Deklariranje na iterator
   vector<int>::iterator i;
   int j,m;
   cout << "Bez iteratori = ";
   // Pristap do elementite bez koristenje na iteratori
   for (j = 0; j < n; ++j) {
      cout << v[j] << " ";
   }
   cout << "\n so iteratori = ";
   // Pristap do elementite so koristenje na iteratori
   for (i = v.begin(); i != v.end(); ++i) {
      cout << *i << " ";
   }
```

```
    // dodavanje na element na vektorot
    cout<<endl;
    cout<<"vnesi element"<<endl;
    cin>>m;
    v.push_back(m);
    cout << "\nBez iteratori = ";
    // Pristap do elementite bez koristenje na iteratori
    for (j = 0; j<=n; ++j) {
       cout << v[j] << " ";
    }
    cout << "\nso iteratori = ";
    // Pristap do elementite so koristenje na iteratori
    for (i = v.begin(); i != v.end(); ++i) {
       cout << *i << " ";
    }
    return 0;
}
```

advance() - оваа функција се користи за да ја зголеми позицијата на итераторот до наведениот број во аргументот на функцијата.

Пр.
```
#include <iostream>
#include <vector>
using namespace std;

int main()
{
    int n,l,k;
    vector <int> v;
        cout<<"vnesi broj na elementi"<<endl;
    cin>>n;
    for(int i=0;i<n;i++)
    {
       cout<<"vnesi element"<<endl;
       cin>>l;
       v.push_back(l);
    }
        vector<int>::iterator ptr = v.begin();
    cout<<"vnesi pozicija"<<endl;
    cin>>k;
    advance(ptr, k);
    cout << "Pozicijata na iteratorot po advance e: ";
    cout << *ptr << " ";
    return 0;
}
```

Листи и итератори

Итераторите се особено важни за **листите** бидејќи не постои начин за директен пристап кон елементите од листата. Важно е да се напомене дека итераторите се всушност покажувачи кон елементите од контејнерот.
Така, нека е декларирана листа ls за елементи од тип int.
list<int> ls;
За да ги поминеме елементите од оваа листа, мора на елементите од листата да им пристапиме преку итератор, односно преку покажувач.

```
 list<int>::iterator p;
for(p=ls.begin();p!=ls.end();p++)
  cout<<*p<<endl;
```

Кога станува збор за листата како контејнер и итераторите, постојат методи коишто дозволуваат вметнување елемент во листа и бришење на елемент во листа.
Додавање на елемент x во листата ls пред итераторот p се врши со методата insert:
ls.insert(p,x);
Бришење на елемент од листата ls, кон којшто покажува итераторот p се врши со методата erase:
ls.erase(p);

пр.

```
#include <iostream>
#include <list>
using namespace std;

void print(list<char> ls)
{
    for(list<char>::iterator p=ls.begin();p!=ls.end();p++)
                        cout<<*p<<" ";
    cout<<endl;
}

int main(){
    list<char> ls;
    list <char>::iterator p;
    ls.push_back('o');
    ls.push_back('a');
    ls.push_back('t');
    p=ls.begin();
// p pokazuva na 'o' vo ('o', 'a', 't')
 cout <<" "<< *p<<endl;
 print(ls);
 ls.insert(p, 'c');
 // ls sega izgleda ('c', 'o', 'a', 't') a p seuste pokazuva na 'o'
 cout <<" "<< *p<<endl;
 print(ls);
 ls.erase(p);
```

```
 // p pokazuva na 'o' koesto veke ne e vo ls
 cout <<" "<< *p<<endl;
 print(ls);
 //za odstranuvanje na prviot element od listata ls
 ls.erase(ls.begin());
 print(ls);
 return 0;
}
```

Забелешка:

Со вклучување на директивата
#include<algorithm>
и употреба на итераторите и векторите можно е ефикасно сортирање преку користење на методата sort.

Пр.

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

//Funkcija za pecatenje na elementite od vektorot a
void print( vector<int> a){
   for(vector<int>::iterator ai=a.begin(); ai!=a.end(); ++ai)
   cout << *ai << " ";
   cout << endl;
}

int main()
{
   int n,i,k;
   //Deklarirame vektor a
   vector<int> a;
   cout<<"vnesi broj na elementi"<<endl;
   cin>>n;
   cout<<"vnesi gi elementite"<<endl;
   for(i=0;i<n;i++)
   {
      cin>>k;
      a.push_back(k);
   }
  //Elementite od vektorot se
   print(a);
  /*Gi sortirame elementite od vektorot, preku povikuvanje na metodata sort od
   STL <algorithm> bibliotekata*/
   sort( a.begin(), a.end() );
   //Elementite se podredeni
```

```
    print(a);
    return 0;
}
```

Прашања поврзани со наставните единици може да се испраќаат на email:
anetastojceska@gmail.com