

-reverzni vektor

```
void ReverseVector (vector<int> &v)
{
    vector<int> temp=v;
    for(unsigned int i=0, unsigned int j=temp.size()-1; i<v.size(); i++, j++)
        v[i]=temp[j];
}
```

-korisnik unosi brojeve koje spremamo u listu; brojeve zatim prepisemo u dinamički niz A; sortirati niz, ispisati sadržaj, dealocirati.

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

void main (void)
{
    double d;
    list<double> l;
    list<double>::iterator iter;

    int brojac(0);
    while(cin>>d)
    {
        l.push_back(d);
        brojac++;
    }
    double *A=new double[brojac];
    int i=0;
    for(iter=l.begin(); iter!=l.end(); iter++, i++)
        A[i]=*iter;

    double temp;
    for(i=0; i<brojac; i++)
        for(j=0;j<i; j++)
            if (A[i]<A[j])
            {
                temp=A[j];
                A[j]=A[i];
                A[i]=temp;
            }
    for(i=0; i<brojac; i++)
        cout<<A[i]<<" ";

    delete A;
}
```

-sadržaj jedne datoteke se kopira u drugu; iz svake linije je u zagradama broj znakova linije; imena datoteka korisnik zadaje u kom. liniji.

```
#include<iostream>
#include<fstream>
#include<string>
using namespace std;

void main(void)
{
    string s1, s2, s3;
    cout<<"Unesite ime izvorisne datoteke";
    cin>>s1;
    cout<<"Unesite ime odredisne datoteke";
    cin>>s2;
    ifstream(s1.c_str());
    ofstream(s2.c_str());
    do
    {
        getline(d1,s3,'\n');
        d2<<s3<<"(""<s3.size()<<"")"<<endl;
    } while (!d1.eof());
    d1.close();
    d2.close();
}
```

-iz stringa s, dobivamo ekvivalentnu numericku vrijednost;

a) pomocu pokazivaca ASCIIZ
b) pomocu objekta klase
a) double StrToInt (char *s)
return (atoi(s))

b) double StrToInt (string s)
return (atoi(s.c_str()));

-s tipkovnice unosimo brojeve u listu; prekida se na nula; pozitivni na pocetak, negativni na kraj; suma pozitivnih, suma negativnih.

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

void main(void)
{
    double x(0), sumapoz(0), sumaneg(0);
    list<double> l;
    while(cin>>x)
    {
        if (x==0)
            break;
        else if (x<0)
        {
            l.push_front(x);
            sumapoz=sumapoz+x;
        }
        else
        {
            l.push_back(x);
            sumaneg=sumaneg+x;
        }
    }
    cout<<"suma pozitivnih"<<sumapoz<<endl;
    cout<<"suma negativnih"<<sumaneg<<endl;
}
```

-kumulativna suma kod vektora
void KumulativnaSuma (vector<int> &v)
{
 for(unsigned int i=1; i<v.size(); i++)
 v[i]=v[i]+v[i-1];
}

-kopiranje datoteka; riječ "node" se zamjenjuje sa "cvor".

```
#include<iostream>
#include<fstream>
#include<string>
using namespace std;

void main(void)
{
    string s1,s2,s3;
    cout<<"Unesi ime izvorisne datoteke";
    cin>>s1;
    cout<<"Unesi ime odredisne datoteke";
    cin>>s2;
    ifstream file1(s1.c_str());
    ofstream file2(s2.c_str());
    int i;
    static const basic_string<char>::size_type npos=-1;
    do
    {
        getline(file1,s1,'\n');

        while ((i=s3.find("node")) != npos)
            s3.replace(i,4,"cvor");
        file2<<s3<<endl;
    } while (!file1.eof());
    file1.close();
    file2.close();
}
```

-funkcija vraća string koji predstavlja literalni zapis:

a) cjelobrojne vrijednosti
b) realne vrijednosti
c) logicke vrijednosti
a) string ToString(int x)

```
{
    char s[20];
    itoa(x,s,10);
    return s;
}
```

b) string ToString(double x)
{
 char s[20];
 _gcvt(x,5,s);
 return s;

c) string ToString(bool b)
{
 if (b) return "true";
 else return "false";
}

-stog; implementacija fja push, pop, top. Main: upis, ispis
#include<vector>
#include<string>

```
class StringStog
{
    vector<string> m_stog;
    public:
        void push(const string& str)
        {
            m_stog.push_back(str);
        }
        void pop()
        {
            m_stog.pop_back();
        }
        void top (string &str)
        {
            str=m_stog[m_stog.size()-1];
        }
        bool empty() const { return m_stog.empty();}
        int size() const { return m_stog.size();}
```

main:

```
void main (void)
{
    StringStog stog;
    string s;
    string str;
    do
    {
        getline(cin, s, '\n');
        if (s.size() != 0)
            stog.push(s);
    } while (s.size() != 0);
    if (stog.empty())
        cout<<"Nije izvršen unos"<< endl;
    else
    {
        cout<<"Uneseno je " <<stog.size()<<"stringova."<<endl;
        cout<<"Uneseni su sljedeci stringovi:"<<endl;

        while (!stog.empty())
        {
            stog.top(str);
            cout<<str<<" ";
            stog.pop();
        }
    }
}
```

-RAZLOMAK:

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

```
class Razlomak
{
private:
    int m_brojnik;
    int m_nazivnik;

public:
```

Razlomak() : m_brojnik(0), m_nazivnik(1) {}

Razlomak(int brojnik, int nazivnik=1)

```
{
    m_brojnik=brojnik;
    m_nazivnik=nazivnik;
}
```

int Brojnik() const { return m_brojnik;}
int Nazivnik() const { return m_nazivnik; }

void Brojnik(int br) { m_brojnik=br; }
void Nazivnik(int naz) { m_nazivnik=naz; }

};

int NZM (int a, int b)

```
{
    int t;
    while(b!=0)
    {
        t=b;
        b=a%b;
        a=t;
    }
    return a;
}
```

Razlomak operator*(const Razlomak &r, const Razlomak &s)

```
{
    int a=r.Brojnik();
    int b=r.Nazivnik();
    int c=s.Brojnik();
    int d=s.Nazivnik();
```

int djelitelj=NZM(a*d,b*c);
return Razlomak((a*c)/djelitelj,(b*d)/djelitelj);
}

Razlomak operator/(const Razlomak &r, const Razlomak &s)

```
{
    int a=r.Brojnik();
    int b=r.Nazivnik();
    int c=s.Brojnik();
    int d=s.Nazivnik();
```

int djelitelj=NZM(a*d,b*c);
return Razlomak((a*d)/djelitelj,(b*c)/djelitelj);
}

Razlomak operator+(const Razlomak &r, const Razlomak &s)

```
{
    int a=r.Brojnik();
    int b=r.Nazivnik();
    int c=s.Brojnik();
    int d=s.Nazivnik();
```

int djelitelj=NZM((a*d)+(c*b), b*d);
return Razlomak(((a*d)+(c*b))/djelitelj, (b*d)/djelitelj);
}

Razlomak operator-(const Razlomak &r, const Razlomak &s)

```
{
    int a=r.Brojnik();
    int b=r.Nazivnik();
    int c=s.Brojnik();
    int d=s.Nazivnik();

    int djelitelj=NZM((a*d)-(c*b), b*d);
    return Razlomak(((a*d)-(c*b))/djelitelj, (b*d)/djelitelj);
}
```

ostream& operator<< (ostream &out, const Razlomak &s)

```
{
    out<<s.Brojnik()<<"/"<<s.Nazivnik();
    return out;
}
```

istream& operator>> (istream &in, Razlomak &r)

```
{
    int b,n;
    char ch;
    if (in>>b>>ch>>n)
    {
        r.Brojnik(b);
        r.Nazivnik(n);
    }
    return in;
}
```

-s tipkovnice unosimo niz imena (string);

unos završava kad unesemo prazni string; imena unosimo u listu i to tako da ona s velikim pocetnim slovom unesemo na pocetak, inatje na kraj; ispisati imena s velikim pocetnim slovom.

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

void main(void)

```
{
    string Ime;
```

list<string> L;

```
do
{
    getline(cin, Ime, '\n');
    if (Ime[0]!='A' && Ime[0]!='Z')
        L.push_front(Ime);
    else
        L.push_back(Ime);
}
```

} while (Ime.size()!=0);

cout<<"Imena koja pocinju velikim slovom:"<<endl;

list<string>::iterator iter;

for(iter=L.begin(); iter!=L.end(); iter++)

```
{
    Ime=*iter;
    if(Ime[0]!='A' && Ime[0]!='Z')
        cout<<Ime<<endl;
}
```

}

-funkcija vraca vektor int elementata;
argument je string cijij se clanovi pretvaraju u ASCII
vrijednosti AKO su slova u pitanju.

```
vector<int> KopirajStringVektor (string s)
{
    vector<int> v;

    for(unsigned int i=0;i<s.size();i++)
    {
        int a=(int)s[i];

        if ( (a>=65 && a<=90) || (a>=97 && a<=122))
            v.push_back(a);
    }
    return v;
}
```

-funkcija vraca listu int elementata;
argument je string cijij se clanovi pretvaraju u ASCII
vrijednosti AKO su slova u pitanju.

```
list<int> KopirajStringLista (string s)
{
    list<int> l;

    for(unsigned int i=0; i<s.size(); i++)
    {
        int a=(int)s[i];
        if ( (a>=65 && a<=90) || (a>=97 && a<=122))
            l.push_back(a);
    }
    return l;
}
```

-reverse lista

```
template <class T> void OkreniListu (list<T> &var)
{
    var.reverse();
}
```

-sort liste

```
list<int> SortirajListu(list<int> var);
{
    var.sort();
    return var;
}
```

-reverse stog

```
template <class T> void OkreniStog(stack<T> &var)
{
    stack<T> s;

    while(!var.empty())
    {
        s.push(var.top());
        var.pop();
    }
    var=s;
}
```

-elementi se skidaju sa stoga i kopiraju u red

```
void KopirajStogRed(stack<int> var1, queue<int> &var2)
{
    while (!var1.empty())
    {
        var2.push(var1.top());
        var1.pop();
    }
}
```

-sadržaj jedne datoteke se kopira u drugu,
NE kopiraju se samoglasnici.

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

void main(void)
{
    ifstream a("prva.txt");
    ofstream b("druga.txt");

    char ch;

    while (a.get(ch))
    {
        if (ch!='a' && ch!='B' ... )
            b<<ch;
    }
    a.close();
    b.close();
}
```

-uneseni brojevi se upisuju u datoteku;
unos zavrsva na neki karakter;
sadržaj datoteke je broj, pa zarez pa zbroj tog i prethodnog;
u prvoj liniji samo uneseni broj.

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

void main(void)
{
    ofstream file("datoteka.txt");

    int i1(0);
    int i2(0);

    cin>>i2;
    file<<i2<<endl;
    i1=i2;

    while (cin>>i2)
    {
        file<<i2<<" "<<i2+i1<<endl;
        i1=i2;
    }

    file.close();
}
```

-20 imena (string) se unose u datoteku;
unese se ime, pa razmak, pa broj slova imena.

```
#include<iostream>
#include<fstream>
#include<string>
using namespace std;

void main(void)
{
    string lme;

    ofstream a("dat.txt");

    for(int i=0;i<20;i++)
    {
        getline(cin,lme,"n");
        a<<lme<<" "<<lme.size()<<endl;
    }
    a.close();
}
```

-funkcija vraca sortirani stog int elemenata;
argument fje je nesortirana lista int elemenata.

```
stack<int> SortirajListu (list<int> var)
{
    stack <int> s;
    list<int> ::iterator iter;

    var.sort();

    for(iter=var.begin(); iter!=var.end(); iter++)
    {
        s.push(*iter);
    }
    return s;
}
```

-fja vraca sortirani red int elemenata;
argument fje je nesortirana lista int elemenata.

```
queue<int> SortirajListu (list<int> var)
{
    queue<int> q;
    list<int>::iterator iter;

    var.sort();

    for(iter=var.begin();iter!=var.end(); iter++)
        q.push(*iter);

    return q;
}
```

-elementi stoga koji se fji predaje kao argument,
se kopiraju u red koji se vraca iz fje.

```
queue<int> KopirajStogRed (stack<int> var1)
{
    queue<int> q;

    while(!var1.empty())
    {
        q.push(var1.top());
        var1.pop();
    }
    return q;
}
```

-unos teksta u datoteku;
sva mala slova pretvaramo u velika;
velika u mala.
ostalo ostaje isto.

```
#include<iostream>
#include<fstream>
#include<string>
using namespace std;

void main(void)
{
    string s;

    ofstream file("dat.txt");

    do
    {
        getline(cin,s,"n");
        for(unsigned int i=0; i<s.size(); i++)
        {
            if (s[i]>='a' && s[i]<='z')
                file<<(char)(s[i]-(a-'A'));
            else if (s[i]>='A' && s[i]<='Z')
                file<<(char)(s[i]+(a-'A'));
            else
                file<<s[i];
        }
        file<<endl;
    } while (s.size() !=0);
}
```

-sadržaj jedne datoteke kopiramo u drugu;
na kraju svake linije u zagradama je broj slova.

```
#include<fstream>
#include<string>
using namespace std;

void main(void)
{
    ifstream file1("datoteka1.txt");
    ofstream file2("datoteka2.txt");

    string s;

    do
    {
        getline(file1, s,"n");
        file2<<s<<"(""<<s.size()<<")"<<endl;
    } while (!file1.eof());

    file1.close;
    file2.close;
}
```

-ako su vektor1 i vektor2 iste velicine,
zbarajamo ih u novi vektor3;
inatje vracamo prazan vektor3.

```
template <class T> vector<T> ZbrojiVektore(vector<T> var2,
vector<T> var3)
{
    vector<T> v;

    if (var2.size() != var3.size())
        return v;
    else
    {
        for(unsigned int i=0; i<var2.size(); i++)
        {
            v.push_back(var2[i]+var3[i]);
        }
        return v;
    }
}
```

-lista se duplicira;
originalna se i vraca;
iza prvog se dodaje zadnji element;
iza drugog predzadnji itd.

```
void DuplaLista(list<int> &var1);
{
    list<int> l;
    list<int>::iterator iter1, iter2;

    iter2=var1.end();
    iter2--;
    for(iter1=var1.begin();iter1!=var1.end(); iter1++,iter2--)
    {
        l.push_back(*iter1);
        l.push_back(*iter2);
    }
    var1=l;
}
```

-preopterećenje operatora < za stand. klasu stack

```
bool operator< (stack<int> &prvi; stack<int> &drugi)
{
    int p(0), d(0);

    if (prvi.size() > drugi.size())
        return true;
    else if (prvi.size() < drugi.size())
        return false;
    else
    {
        while(!prvi.empty())
        {
            if (prvi.top() > drugi.top())
                p++;
            else if (prvi.top() < drugi.top())
                d++;

            prvi.pop();
            drugi.pop();
        }

        if (p>d)
            return true;
        else return false;
    }
}
```

-elementi stoga var1, i elementi reda var2 se
naizmjenice kopiraju u listu koja se vraca iz fje;
stog i red mogu imati različit broj elemenata;
u tom slučaju kopira se do kraja.

```
template <class T> list<T> Kopiraj (stack<T> var1, queue<T>
var2)
{
    list<T> L;

    while(!var1.empty() && var2.empty())
    {
        if (!var1.empty())
        {
            L.push_back(var1.top());
            var1.pop();
        }

        if (!var2.empty())
        {
            L.push_back(var2.front());
            var2.pop();
        }
    }
    return L;
}
```