–FESB –

Objektno orjentirano programiranje (720) i Programiranje II (750)

- riješeni ispitni primjeri -

- dipl. ing. Špiro Krasić -
skrasic@gmail.com

– Split, prosinac 2005 –

Sadržaj

05.07.2002.

Zadatak 1.

Napišite koji ispis generira sljedeći program

```
int FunctionByValue(int i) { return ++i; }
int FunctionByRef(int &ir) { return ++ir; }
int FunctionByAddr(int *ia) { return ++(*ia); }

void main() {
     int i = 0;             cout << i << endl;
     FunctionByRef(i);      cout << i << endl;
     FunctionByValue(i);    cout << i << endl;
     FunctionByAddr(&i);    cout << i << endl;
 }
```

Rješenje:

0
1
1
2
Press any key to continue

```
/*   Kad se varijabla prenosi u funkciju po vrijednosti (by value), sve promjene
te varijable bit će vidljive samo unutar funkcije zato što kompajler tada stvara
kopiju te varijable i u funkciji radi sa tom kopijom. Po izlasku iz funkcije
varijabla će poprimiti vrijednost koju je imala i prije poziva funkcije.

Ako se varijabla prenosi 'by ref' ili 'by addr', u funkciji se radi sa originalnom
varijablom (ne radi se kopija) te sve promjene varijable bit će sačuvane i po
izlasku iz funkcje. Prijenos 'by ref' i 'by addr' je brži ali i osjetljiviji na
pogreške od prijenosa 'by value'.   */
```

Zadatak 2.

Napišite funkciju čiji prototip glasi:

```
void ReverseVector(vector<int> &v)
    // pre: vektor v sadrži N cijelih brojeva
    // post: elementi od v su u reverznom redoslijedu
```

i koja ulaznom vektoru v mijenja redoslijed elemenata tako da oni budu u reverznom redoslijedu u odnosu na početni redoslijed. Primjerice,

```
Prije poziva ReverseVector(a)        Poslije poziva ReverseVector(a)

+----+----+----+----+----+           +----+----+----+----+----+
| 61 | 34 | 18 | 99 | 73 |           | 73 | 99 | 18 | 34 | 61 |
+----+----+----+----+----+           +----+----+----+----+----+
```

Rješenje:

```
void ReverseVector(vector<int> &v)
{
      vector<int> temp=v;   // vektor temp je jednak ulaznom vektoru v

      for(int i=0, int j=temp.size()-1; i<v.size(); i++,j--)
      {
            v[i]=temp[j];
      }
}
```

Zadatak 3.

Deklarirana je klasa Point, kojoj privatni članovi `int m_x, m_y` opisuju poziciju točke u ravnini.

```
class Point {
      int m_x, m_y;
public:
     Point():m_x(0), m_y(0) {}
     Point(int x, int y):m_x(x), m_y(y) {}
     int X() {return m_x;};
     int Y() {return m_y;};
};
```

Klasi Trokut privatni su članovi tri objekta klase Point : `m_p1, m_p2 i m_p3,` koji određuju koordinate trokuta.

```
class Trokut {
       Point m_p1,m_p2,m_p3;
      public:
            // definiraj funkcije
            Trokut(Point p1, Point p2, Point p3);
            double Opseg();
};
```

Definirajte konstruktor Trokut() i metodu Opseg() kojom se računa opseg trokuta, tako da se može realizirati program:

```
int main()
{
    Trokut tr(Point(7,1),Point(1,1), Point (4,3));
    cout <<"Opseg =" << tr.Opseg() <<endl;
    return 0;
}
```

koji nakon izvršenja daje ispis: `Opseg =13.2111`

Rješenje:

```
/*  Ako definicije konstruktora i funkcije pišemo unutar klase Trokut:   */

Trokut(Point p1, Point p2, Point p3)
{
      m_p1 = p1;
      m_p2 = p2;
      m_p3 = p3;
}

double Opseg()
{
      double d1(0), d2(0), d3(0);

      d1 = sqrt(pow((m_p1.X() - m_p2.X()),2) + pow((m_p1.Y()- m_p2.Y()),2));
      d2 = sqrt(pow((m_p1.X() - m_p3.X()),2) + pow((m_p1.Y()- m_p3.Y()),2));
      d3 = sqrt(pow((m_p2.X() - m_p3.X()),2) + pow((m_p2.Y()- m_p3.Y()),2));

      return d1+d2+d3;
}
```

```
/*  Ako definicije konstruktora i funkcije pišemo izvan klase Trokut:   */

Trokut::Trokut(Point p1, Point p2, Point p3)
{
      m_p1 = p1;
      m_p2 = p2;
      m_p3 = p3;
}

double Trokut::Opseg()
{
      double d1(0), d2(0), d3(0);

      d1 = sqrt(pow((m_p1.X() - m_p2.X()),2) + pow((m_p1.Y()- m_p2.Y()),2));
      d2 = sqrt(pow((m_p1.X() - m_p3.X()),2) + pow((m_p1.Y()- m_p3.Y()),2));
      d3 = sqrt(pow((m_p2.X() - m_p3.X()),2) + pow((m_p2.Y()- m_p3.Y()),2));

      return d1+d2+d3;
}

/*  Ako definicije konstruktora i funkcije pišemo izvan klase Trokut tada je
unutar klase potrebno navesti deklaraciju konstruktora i funkcije:    */

class Trokut
{
private:
          .
          .
          .
public:
      Trokut(Point p1, Point p2, Point p3);  // deklaracija konstruktora
      double Opseg();                        // deklaracija funkcije Opseg()
};
```

Zadatak 4.

Napišite program u kojem :

1. korisnik unosi proizvoljan broj realnih brojeva . Te brojeve treba spremiti u kontenjer tipa list<double> .
2. Nakon završenog unosa treba oformiti dinamički niz A, u kojeg treba upisati sve brojeve iz liste.
3. Konačno treba sortirati taj niz, ispisati njegov sadržaj i dealocirati niz A.

Rješenje:

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

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

	int brojac(0);  // ovaj broj će nam govoriti koliko je brojeva korisnik unio
	while(cin >> d)
	{
		l.push_back(d);
		brojac++;
	}

	double *A = new double[brojac];   // dinamički stvoren niz A
    int i=0;
	for(iter=l.begin(); iter!=l.end();iter++,i++)
	    A[i]=*iter; //dereferencira iter i dobivena vrijednost se sprema u niz A

	double temp;
	for(i=0; i<brojac; i++)  // algoritam za sortiranje 'bubble sort'
		for(int 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++)  // ispisiva sadržaj niza
		cout << A[i] << " ";   // dealocira niz A

	delete A;


}

/*

Kroz listu  list<doube> l 'šetamo' pomoću list<double>::iterator iter objekta koji
je inicijaliziran na l.begin() i uvećavamo ga sve dok je različit od l.end().
Dereferencirajući objekt iter očitavamo vrijednost elementa koji se nalazi na toj
poziciji liste.

*/
```

Zadatak 5.

Napišite program kojim se sadržaj jedne tekstualne datoteke kopira u drugu datoteku, ali tako da se iza svake linije u zagradama zapiše koliko ima znakova u liniji. Ime izvorne i odredišne datoteke zadaje korisnik u komandnoj liniji.
Primjerice ako je u izvornoj datoteci linija sadržaja

```
hello world!
```

tada u odredišnoj datoteci mora biti linija:

```
hello world! (12)
```

Rješenje:

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

void main(void)
{
	string s1,s2,s3;

	cout << "Unesite ime izvorisne datoteke: ";
	cin >> s1;
	cout << endl << "Unesite ime odredisne datoteke: ";
	cin >> s2;

	ifstream d1(s1.c_str());
	ofstream d2(s2.c_str());

	do
	{
		getline(d1,s3,'\n');
		d2 << s3 << " " << s3.size() << endl;
	}while(!d1.eof());  // dok ne dođe do kraja datoteke

	d1.close();
	d2.close();
}
```

20.09.2002.

Zadatak 1.

Napišite dvije verzije potprograma:

```
double StrToInt(char *s);
double StrToInt(string s);
```

pomoću kojeg se iz stringa s, koji sadrži niz numeričkih znakova, dobije ekvivalentna numerička realna vrijednost. Prvi oblik funkcije kao parametar prima pokazivač na niz znakova (ASCIIZ), a drugi oblik prima objekt tipa standardne klase string.

Rješenje:

```
double StrToInt(char *s)
{
	double d(0);
	d = atof(s);
	return d;
}

double StrToInt(string s)
{
	double d(0);
	d = atof(s.c_str());
	return d;
}

/*

Funkcija c_str() pretvara 'string s' u 'char *s'.

*/
```

Zadatak 2.

Napišite program kojim se s tipkovnice unosi niz brojeva. Unos završava kada se otkuca nula broj. Brojeve treba unositi u listu tako da se pozitivni brojevi ubacuju na početak liste, a negativni brojevi na kraj liste. Kostur programa je:

```
int  main()
{
      double x;            // broj koji se unosi
      list<double> v ;   // lista u kojem pamtimo unesene brojeve
      double sumapozitivnih, sumanegativnih;
      // Ponavljaj:
      //    1. Dobavi broj x s tipkovnice
      //    2. Ako je broj x jednak nuli
      //          prekini unos
      //       Ako je broj x pozitivan
      //            spremi njegovu vrijednost na početku liste.
      //       Ako je broj x negativan
      //            spremi njegovu vrijednost na kraju liste.
      // Sumiraj odvojeno sve pozitive i sve negativne brojeve pa
      // ispiši te vrijednosti obje sume

}
```

Rješenje:

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

void main(void)
{
      double x(0);
      list<double> v;
      double sumapozitivnih(0), sumanegativnih(0);

      do
      {
            cin >> x;
            if(x>0) v.push_front(x);
            else v.push_back(x);
      }while(x);  // dok je uneseni broj razlicit od 0

      list<double>::iterator iter;

      for(iter=v.begin(); iter!=v.end(); iter++)
      {
            if(*iter>0) sumapozitivnih += *iter;
            else sumanegativnih += *iter;
      }

      cout << "Suma pozitivnih: " << sumapozitivnih << endl;
      cout << "Suma negativnih: " << sumanegativnih << endl;

}

/*    Kroz listu se ne može 'šetati' pomoću indexa (kao kroz vector ili niz). Zato
se koristi iterator.    */
```

Zadatak 3.

Napišite funkciju čiji prototip glasi:

```
void KumulativnaSuma(vector<int> &v)

// pre: vektor v sadrži N cijelih brojeva
// post: elementi od v predstavljaju sumu prethodnih elemenata
```

i koja ulaznom vektoru v mijenja vrijednost elemenata tako da nakon izvršenja oni predstavljaju kumulativnu sumu prethodnih elemenata.

Primjerice,

Prije poziva KumulativnaSuma(v)

v	1	3	7	2	3

Poslije poziva KumulativnaSuma(v)

1	4	11	13	16

Rješenje:

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

```
/*

v.size() vraća broj tipa 'unsigned int'. Zbog toga je i brojač 'i' tog tipa. Ovo
nije nuzno ali je preporučljivo. Ako je brojač tipa 'int' kompajler javi warning
da se uspoređuju dva broja različitih tipova.

'unsigned int' broj može biti samo pozitivan, dok 'int' moze biti i negativan

*/
```

Zadatak 4.

Deklarirana je klasa Point2D kojom se opisuje položaj točke u 2D prostoru

```
class Point2D
{
 public:
      Point2D(): m_x(0),m_y(0);
      void SetX(int x) {m_x = x; }
      void SetY(int y) {m_y = y; }
      int GetX() {return m_x; }
      int GetY(){return m_y;}
 protected:
      int m_x, m_y;
};
```

Koristeći klasu Point2D i pravila naslijeđivanja definirajte klasu Point3D, pomoću koje se opisuje položaj točke u trodimenzionalnom prostoru (x,y,z).

```
class Point3D:public Point2D
{
 public:
 ........
 ........
 protected:
      int m_z;
}
```

a) Definirajte potrebne public metode kojima se mijenja i očitava položaj točke (SetZ, GetZ)
b) Napišite konstruktor, kojim se ujedno inicijalizira položaj točke u x=0 i y=0, z=0.
c) Definiraj `operator=` za klasu Point3D.
d) Definiraj kopirni konstruktor klase Point3D.

Rješenje:

```
class Point3D:public Point2D
{
public:
      Point3D()   // default konstruktor
      {
            m_x = 0;
            m_y = 0;
            m_z = 0;
      }
      Point3D(const Point3D &p)  // kopirni konstruktor
      {
            m_x = p.m_x;
            m_y = p.m_y;
            m_z = p.m_z;
      }
      Point3D& operator =(const Point3D &p)
      {
            m_x = p.m_x;
            m_y = p.m_y;
            m_z = p.m_z;
            return *this;
      }
      void SetZ(int x) {m_z = x;}
      int GetZ() {return m_z;}
protected:
      int m_z;
};
```

Zadatak 5.

Napišite program kojim se sadržaj jedne tekstualne datoteke kopira u drugu datoteku, ali tako da se u svakoj liniji teksta u kojoj se pronađe riječ "node", izvrši zamjena te riječi s rječju "cvor". Ime izvorne i odredišne datoteke zadaje korisnik u komandnoj liniji.

Primjerice, ako je u izvornoj datoteci linija sadržaja

```
if(node.next == NULL)
```

tada u odredišnoj datoteci mora biti linija:

```
if(cvor.next == NULL)
```

Rješenje:

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

void main(void)
{
	string s1,s2,s3;

	cout << "Unesite ime izvorisne datoteke: ";
	cin >> s1;
	cout << endl << "Unesite 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,s3,'\n');

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

	}while(!file1.eof());

	file1.close();
	file2.close();
}
/*	Vanjska 'do while' petlja prolazi kroz sve retke datoteke. Unutarnja
'while' petlja pretražuje trenutni redak i potrebna je zbog toga što u jednom
retku može biti više riječi 'node'. Da nema ove petlje, samo prvi 'node' u retku
bi bio zamjenjen sa 'cvor'.

s3.find("node") vraća index prvog znaka pronađenog stringa. Ako traženi string
nije pronađen fnkcija vraća vrijednost -1.

s3.replace(i,4,"cvor") zamjenjuje 4 karaktera u stringu s3 sa stringom 'cvor'
počevši sa indexom 'i'	*/
```

04.02.2003.

Zadatak 1.

Napišite što će biti ispisano nakon izvršenja ovih programa:

a)
```
#include <iostream>

void main(void)
{
      int y = 0;
      std::cout <<  y++;
      if(y)
      {
          int y = 5;
          std::cout << y;
      }
      std::cout << y;
}
```

b)
```
#include <iostream>
int y = 0;
 void main(void)
 {

    cout << y;
    if(++y)
    {
       int  y = 5;
       std::cout << ++y;
    }
     std::cout << y;
 }
```

Rješenje:

---

a)

051Press any key to continue

b)

061Press any key to continue

```
/*

Lokalna varijabla y postojana je samo unutar bloka u kojem je stvorena. To je 'if'
blok.
Ako lokalna varijabla ima isto ime kao i neka globalna, tada unutar bloka lokalne
varijable, lokalna varijabla ima veći značaj.

*/
```

Zadatak 2.

Napišite tri verzije funkcijeToString:

```
string ToString(int x);
string ToString(double x);
string ToString(bool b);
```

Funkcija vraća string koji predstavlja literalni zapis a) cjelobrojne vtijednosti, b) realne vrijednosti i c) logičke vrijednosti.

Rješenje:

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

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

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


/*

itoa
=====
- prvi parametar: int broj koji se konvertira
- drugi parametar: char niz u koji će se spremiti rezultat
- treći parametar: int baza broja koji se konvertira (npr. za decimalni broj je
10, za binarni 2 ...)


_gcvt
======
- prvi parametar: double broj koji se konvertira
- drugi parametar: int broj decimalnih mijesta koja će se konvertirati
- treći parametar: char niz u koji će se spremiti rezultat

*/
```

Zadatak 3.

Deklaracijom klase StringStog izvršena je specifikacija za kontenjer objekata tipa string. Objektima se pristupa po principu LIFO. Implementirajte funkcije push, top i pop.

```
#include <vector>
#include <string>

class StringStog
{
  vector<string> m_stog;

public:
 void push(const string& str); // postavlja string str na vrh stoga
 void pop ();              // skida string s vrha stoga
 void top (string &str );      // postavlja string s vrha stoga u  str

 bool empty()const {return m_stog.empty();} // true ako je stog prazan
 int size()const {return m_stog.size();}   // vraća broj stringova na stogu
};
```

Rješenje:

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

class StringStog
{
      vector<string> m_stog;

public:
      void push(const string& str)  // stavlja 'str' na stog
      {
            m_stog.push_back(str);
      }

      void pop()  // skida element sa vrha stoga
      {
            m_stog.pop_back();
      }

      void top(string &str)   // kopira element sa vrha stoga u 'str'
      {
            str = m_stog[m_stog.size()-1];
      }

      bool empty()const {return m_stog.empty();} // vraća true ako je stog prazan
      int size()const {return m_stog.size();}  // vraća veličinu stoga
};
```

Zadatak 4.

Napišite program za testiranje klase StringStog. Program treba obaviti sljedeće radnje:

1) izvjestiti korisnika da unese proizvoljan broj stringova, zaključno s praznim stringom
2) napisati petlju u kojoj se dobavljaju stringovi u kontenjer tipa StringStog
3) po završenom unosu provjeri da li je stog prazan.
   Ako je stog prazan
   ispisati poruku: "nije izvrsen unos"
   inače
   ispisati koliko je uneseno stringova
   i sadržaj stoga

Rješenje:

```
void main(void)
{
	StringStog stog;
	string s;
	string str;

	do
	{
		getline(cin,s,'\n');

		// posljednji (prazni) string ne želimo staviti na stog
		if(s.size()!=0)
			stog.push(s);

	}while(s.size()!=0);   // dok nije unesen prazan string

	if(stog.empty())
		cout << "Nije izvrsen 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();
		}

		cout << endl;
	}
}

/*

Kod stoga možemo pročitati samo najgornji element. To znači da, kada želimo
pročitati sve elemente stoga moramo skidati elemente jedan po jedan te ih tako
čitati. To znači da čitanjem uništavamo stog.

*/
```

Zadatak 5.

Deklarirana je klasa Razlomak, kojom se definira objekt razlomka određen cjelobrojnim brojnikom i nazivnikom

```
class Razlomak {
            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;}
            // pristupnici
            int Brojnik() ; /*definiraj*/
            int Nazivnik() const {return  m_nazivnik;}
            // mutatori
            void Brojnik(int br); /*definiraj*/
            void Nazivnik(int naz) { m_nazivnik=naz;}
};

//operacije s razlomcima
Razlomak operator*(const Razlomak &r, const Razlomak &s)
{
      int a = r.Brojnik();
      int b = r.Nazivnik();
      int c = s.Brojnik();
      int d = s.Nazivnik();
      return Razlomak(a*c, b*d);
}

Razlomak operator + (const Razlomak &r,const Razlomak &s);
Razlomak operator -(const Razlomak &r,const Razlomak &s)
```

Koristeći prethodnu deklaraciju klase Razlomak,

a) Definiraj funkcije Brojnik, kojima se postavlja i dobavlja vrijednost brojnika.
b) Definiraj `operator+ i operator-` za klasu Razlomak.
c) Za klasu Razlomak definiraj operatore za ispis i unos razlomka u obliku a/b:

```
ostream& operator << (ostream &out, const Razlomak &s);
istream& operator >> (istream &in, Razlomak &r);
```

tako da se programom:

```
      Razlomak r, s;
      cout << "Unesi razlomak u obliku a/b: " << flush;
      cin >> r;
      cout << "Unesi razlomak u obliku a/b: " << flush;
      cin >> s;

      Razlomak suma = r + s;
      Razlomak produkt = r * s;

      cout << r << " + " << s << " = " << suma << endl;
      cout << r << " * " << s << " = " << produkt << endl;
```

dobije ispis:

```
Unesi razlomak u obliku a/b: 3/4
Unesi razlomak u obliku a/b: 7/3
3/4 + 7/3 = 37/12
3/4 * 7/3 = 21/12
```

Rješenje:

```
#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; }  // dobavlja brojnik
	int Nazivnik() const { return  m_nazivnik;}

	void Brojnik(int br) { m_brojnik = br;}   // postavlja brojnik
	void Nazivnik(int naz) { m_nazivnik=naz;}
};
Razlomak operator*(const Razlomak &r, const Razlomak &s)
{
	int a = r.Brojnik();
	int b = r.Nazivnik();
	int c = s.Brojnik();
	int d = s.Nazivnik();
	return Razlomak(a*c, b*d);
}
Razlomak operator+ (const Razlomak &r, const Razlomak &s)
{
	int a = r.Brojnik();
	int b = r.Nazivnik();
	int c = s.Brojnik();
	int d = s.Nazivnik();
	return Razlomak((a*d)+(c*b), b*d);
}
Razlomak operator- (const Razlomak &r, const Razlomak &s)
{
	int a = r.Brojnik();
	int b = r.Nazivnik();
	int c = s.Brojnik();
	int d = s.Nazivnik();
	return Razlomak((a*d)-(c*b), b*d);
}
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;
}
```

02.09.2003.

Zadatak 1.

Što će biti ispisano po izvršenju ovih programa

a)
```
#include <iostream>

#define MAX(a,b) ((a)>(b))?(a):(b)

int main()
{
  int x = 0, y = 2, z;
  if (z = MAX(x,y))
  std::cout << "Vrijednost z je: " << z;
  return 0;
}
```

b)
```
#include <iostream>
#define LI "ni"
#define IMATE "Nemam"
#define KUNU " lipe\n"

int main(void)
{
  int i = 15;
  std::cout << IMATE << LI << KUNU;
  return 0;
}
```

Rješenje:

---

a)

Vrijednost z je: 2Press any key to continue

```
/*

#define MAX(a,b) ((a)>(b))?(a):(b)

ako je tvrdnja točna vraća 'a' inače vraća 'b'. */
```

b)

Nemamni lipe
Press any key to continue

Zadatak 2.

Napišite koji ispis na ekranu nastaje nakon izvršenja segmenta programa:

```
string word = "omiljeni lik";
int num = 8;

cout << num/2 << "+" << num % 4 << endl;
cout << 8 << "8" << endl << 8 + 10 * num << endl;
cout << word.substr(word.find("mi"),3) << "ko" << endl;
```

Rješenje:

4+0
88
88
milko
Press any key to continue

```
/*

word.substr(word.find("mi"),3) vraća string sadržan u stringu word duljine 3 sa početnim indexom jednakim word.find("mi").

word.find("mi") vraća index prvog znaka traženog stringa 'mi' u stringu 'word'. Ako string nije pronađen funkcija vraća -1.


*/
```

Zadatak 3.

Deklaracijom klase NumberStack izvršena je specifikacija za kontenjer objekata tipa double. Objektima se pristupa po principu LIFO.

```
#include <vector>

class NumberStack {           // stog brojeva
public:
 void push(const double num);  // postavlja "num" na vrh  stoga
 void pop (double &num );      // skida broj s vrha stoga u referencu num
 bool empty()const {return m_stack.empty();} // true ako je stog prazan
 int size()const {return m_Stack.size();}   // vraća broj elemenata stoga
private:
   vector<double> m_Stack;
};
```

Implementirajte funkcije push() i pop().

Rješenje:

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

class NumberStack
{
public:
      void push(const double num)   // stavlja num na stog
      {
            m_Stack.push_back(num);
      }
      void pop (double &num )    // skida element sa stoga i sprema ga u num
      {
            num = m_Stack.back();
            m_Stack.pop_back();
      }
      bool empty()const {return m_Stack.empty();}
      int size()const {return m_Stack.size();}

private:
      vector<double> m_Stack;
};
```

Zadatak 4.

Napišite program za testiranje klase NumberStack. Program treba obaviti sljedeće radnje:

4) Izvjestiti korisnika da unese proizvoljan broj brojeva
5) Napisati petlju u kojoj se dobavljaju brojevi u kontenjer tipa NumberStack
   Prekini petlju kada nije unesen broj
6) Po završenom unosu provjeri da li je stog prazan.
   Ako je stog prazan
   ispisati poruku: "nije izvrsen unos"
   inače
   ispisati koliko je uneseno brojeva
   i sadržaj stoga

Rješenje:

```
void main(void)
{
	NumberStack n;
	double d;
	while(cin >> d)  // dok se unose brojevi
	{
		n.push(d);
	}

	if(n.empty())
		cout << endl << "Nije izvrsen unos!" << endl;
	else
	{
		cout << "Velicina stoga: " << n.size() << endl;

		cout << "Sadzaj stoga: ";

		while(!n.empty())
		{
			n.pop(d);
			cout << d << " ";
		}
	}
}

/*

Posljednjom 'while' petljom stog je uništen, ali to je jedini (jednostavni) način
da vidimo sadržaj stoga. Slično vrijedi i za red (queue). Kad želimo pročitati
stog (ili red) može se njegove elemente spremati u neki privremeni objekt kao što
je lista, red ili vector, te iz tog objekta kasnije rekonstruirati stog.

*/
```

Zadatak 5.

Napišite program kojim se s tipkovnice unosi niz imena (string). Unos završava kada se otkuca prazni string. Stringove treba unositi u listu tako da se imena koja počimaju s velikim slovom ubacuju na početak liste, a imena koja započimaju s malim slovom na kraj liste. Na kraju treba ispisati imena koja započimaju s veliki slovom. Kostur programa je:

```
#include <string>
#include <list>

int  main()
{
        string Ime;          // ime koji se unosi

        list<string> L ;   // lista u kojem pamtimo unesene stringove


        // Ponavljaj:
        //          1. Dobavi Ime s tipkovnice
        //          2. Ako je Ime prazan string
        //         prekini unos
        //       Ako Ime započima s velikim slovom
        //          dodaj ga na početak liste L.
        //       Ako Ime započima s malim slovom
        //          dodaj njegovu vrijednost na kraju liste L.
        //     3. Ispiši sva imena koja započimaju s velikim slovom

        return 0;
}
```

Rješenje:

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

int  main()
{
      string Ime;
      list<string> L;
      do
      {
            getline(cin,Ime,'\n');
            if(Ime[0]>='A' && Ime[0]<='Z')  // ako je prvi znak veliko slovo
                  L.push_front(Ime);
            else
                  L.push_back(Ime);
      }while(Ime.size()!=0);  // dok se ne unese prazan string

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

      list<string>::iterator iter;
      for(iter=L.begin(); iter!=L.end(); iter++)
      {
            Ime = *iter;
            if(Ime[0]>='A' && Ime[0]<='Z')  // ako je prvi znak veliko slovo
                  cout << Ime << endl;
      }
      return 0;
}
```

08.02.2005.

Zadatak 1.

Napisati specifikaciju i implementaciju apstraktnog tipa podatka ***kocke*** (deklarirati i definirati klasu koja opisuje kocku). Svi podatkovni članovi klase trebaju biti deklarirani kao **zaštićeni članovi (private)**. Objekti klase ***kocka*** trebaju imati integriranu funkcionalnost proračuna volumena i oplošja kocke (funkcijske članove za proračun volumena i oplošja kocke).

Rješenje:

```
class Kocka
{
private:
	int a;

public:
	void postavi_a(int broj)
	{
		a = broj;
	}

	int dohvati_a(void)
	{
		return a;
	}

	float povrsina()
	{
		return 6*a*a;
	}

	float volumen()
	{
		return a*a*a;
	}
};
```

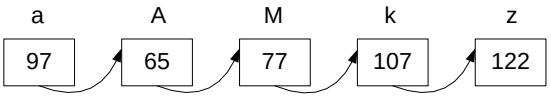
Zadatak 2.

Napisati definiciju funkcije koja vraća vektor *int* elemenata. Kao argument funkcija prima string i zatim karaktere string kopira u vektor na način da u vektor kopira njihove ASCII vrijednosti. Ovo se odnosi samo na mala (a(97) - z(122), A(65) – Z(90)) i velika slova dok se ostali karakteri iz string preskaču. Deklaracija funkcije je:

vektor*<*int*>** KopirajStringVektor(string*** var);

Primjer:

KopirajStringVektor("aA!Mk@z") →



Rješenje:

```
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) // ako je slovo
			v.push_back(a);
	}
	return v;
}
```

Zadatak 3.

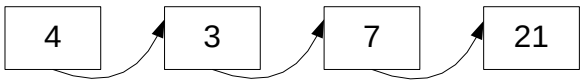
Napisati definiciju (kod) generičke funkcije

template <class T> void OkreniListu(***list*** <T> &var)

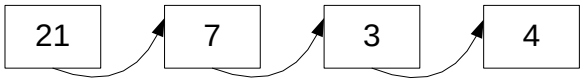
kojom se elementi liste koja se funkciji predaje po referenci kao argumenti poredaju obrnutim redoslijedom (prvi element ide na posljednje mjesto u listi, drugi na pretposljednje, ...).

Primjer:

lista prije poziva funkcije



lista nakon poziva funkcije



Rješenje:

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

/*

Postoji metoda 'reverse' kojom se obrne redosljed elemenata u listi.  :)

*/
```

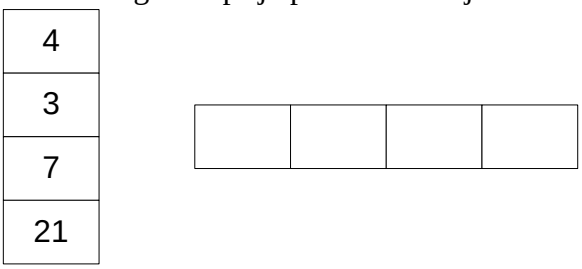
Zadatak 4.

Napisati definiciju (kod) funkciju kojom se elementi stoga sa integer elementima koji se funkcije predaje kao argument kopiraju u prazni red koji se funkciji predaje po referenci kao argument. Deklaracija funkcije je:

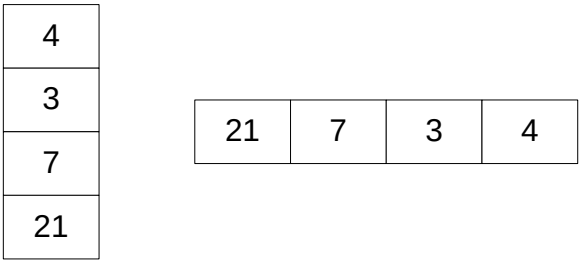
void KopirajStogRed(***stack<int>*** var1, ***queue<int>*** &var2)

Primjer:

stog i red prije poziva funkcije



stog i red nakon poziva funkcije



Rješenje:

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

Zadatak 5.

Napisati program kojim se sadržaj jedne postojeće originalne tekstualne datoteke kopira u novu kreiranu tekstualnu datoteku na način da se iz originalne datoteke u novu kreiranu datoteku ne kopiraju samoglasnici (to vrijedi i za mala i za velika slova). Svi ostali karakteri se kopiraju.

Primjer:

originalna datoteka

```
Danas je utorAk
Sutra je srijeda
PreksUtra je četvrak
```

kopirana datoteka

```
Dns j trk
Str j srjd
Prkstr j čtvrk
```

Rješenje:

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

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

	char ch;

	while(a.get(ch))  // čita zakove jedan po jedan
	{
		// ako pročitani znak nije samoglasnik (ni mali ni veliki)
		if(ch!='a' && ch!='e' && ch!='i' && ch!='o' && ch!='u'&& ch!='A' &&
                                        ch!='E' && ch!='I' && ch!='O' && ch!='U')
			b << ch;
	}

	a.close();
	b.close();
}
```

22.02.2005.

Zadatak 1.

Napisati implementaciju apstraktnog tipa podatka ***Osoba***. Klasa je deklarirana sa:

```
class Osoba
{
  private:
        string ime;
        string prezime;
        int godina_rodjenja;
  public:
        //funkcija postavlja vrijednosti sva tri podatkovna člana
        void postavi_sve(string var1,string var2,int var3);
        //funkcija vraća vrijednost člana klase ime
        string dohvati_ime();
        //funkcija vraća vrijednost člana klase prezime
        string dohvati_prezime();
        //funkcija vraća vrijednost člana klase godina rođenae
        int dohvati_godinu_rodjenja();
};
```

Rješenje:

```
class Osoba

{

private:
      string ime;
      string prezime;
      int godina_rodjenja;

public:
      void postavi_sve(string var1, string var2, int var3)
      {
            ime = var1;
            prezime = var2;
            godina_rodjenja = var3;
      }

      string dohvati_ime()
      {
            return ime;
      }

      string dohvati_prezime()
      {
            return prezime;
      }

      int dohvati_godinu_rodjenja()
      {
            return godina_rodjenja;
      }
};
```

Zadatak 2.

Napisati implementaciju apstraktnog tipa podatka ***Student***. Klasa nasljeđuje klasu ***Osoba***. Klasa je deklarirana sa:

```
class Student:public Osoba
{
  private:
        int godina_upisa;
        int trenutna_godina;
  public:
        //nadređena funkcija postavi_sve funkciji postavi_sve iz klase Osoba postavlja //vrijednost podatkovnih članova
        ime, prezime, godina_rodjenja, //godina_upisa i trenutna_godina
        void postavi_sve(string var1,string var2,int var3,int var4, int var5);
        //funkcija vraća vrijednost člana klase godina_upisa
        int dohvati_godinu_upisa ();
        //funkcija vraća vrijednost člana klase trenutna godina
        int dohvati_trenutnu_godinu();
};
```

Rješenje:

```
class Student:public Osoba
{
private:
      int godina_upisa;
      int trenutna_godina;

public:
      void postavi_sve(string var1, string var2, int var3, int var4, int var5)
      {
            Osoba::postavi_sve(var1, var2, var3);
            godina_upisa = var4;
            trenutna_godina = var5;
      }

      int dohvati_godinu_upisa()
      {
            return godina_upisa;
      }

      int dohvati_trenutnu_godinu()
      {
            return trenutna_godina;
      }
};
```

Zadatak 3.

Napisati definiciju (kod) funkcije preopterećenja operatora << i >> za klasu *Student*. Funkcije su deklarirane sa:

ostream& operator<<(ostream &var1, Student &var2);

istream& operator>>(istream &var1, Student &var2);

Rješenje:

```
ostream& operator<<(ostream &var1, Student &var2)
{
	var1 << "Ime: " << var2.dohvati_ime() << endl;
	var1 << "Prezime: " << var2.dohvati_prezime() << endl;
	var1 << "Godina rodjenja: " << var2.dohvati_godinu_rodjenja() << endl;
	var1 << "Godina upisa: " << var2.dohvati_godinu_upisa() << endl;
	var1 << "Trenutna godina: " << var2.dohvati_trenutnu_godinu() << endl;

	return var1;
}

istream& operator>>(istream &var1, Student &var2)
{
	string s1,s2;
	int i1,i2,i3;

	if(var1 >> s1 >> s2 >> i1 >> i2 >> i3)
	{
		var2.postavi_sve(s1,s2,i1,i2,i3);
	}
	return var1;
}

/* Unutar klase treba dodati deklaracije ovih dviju funkcija kao 'friend' funkcija
da bi iz njih mogli pristupiti privatnim članovima klase. */


friend ostream& operator<<(ostream &var1, Student &var2);

friend istream& operator>>(istream &var1, Student &var2);
```

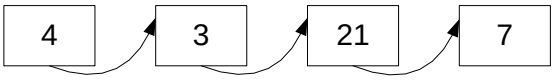
Zadatak 4.

Napisati definiciju (kod) funkciju koja vraća sortiranu listu *int* elemenata. Kao argument funkcija prima nesortiranu listu *int* elemenata. Deklaracija funkcije je:

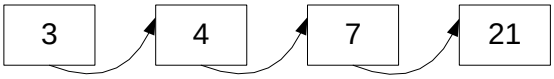
list<int> SortirajListu(***list<int>*** var);

Primjer:

lista *var* koja se funkciji predaje kao argument



lista koja se vraća iz funkcije



Rješenje:

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

/*

Postoji metoda 'sort' kojom se sortira lista.   :)

*/
```

Zadatak 5.

Napisati program kojim se cjelobrojni brojevi *(int)* koje korisnik unosi preko tipkovnice unose u novu kreiranu tekstualnu datoteku. Unos završava kada korisnik unese karakter preko tipkovnice. Sadržaj datoteke u svakoj liniji je uneseni broj, zatim karakter zarez i zatim suma unesenog broja i prethodnog broja. U prvoj liniji nalazi se samo uneseni broj.

Primjer:

korisnikov unos	sadržaj datoteke
3	3
23	23, 26
111	111,134
4	4,115
7	7,13

Rješenje:

```
#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)  // dok se unosi broj
	{
		file << i2 << "," << i2+i1 << endl;
		i1=i2;
	}
	file.close();
}
```

02.04.2005.

Zadatak 1.

Napisati implementaciju apstraktnog tipa podatka ***Student*** specificiranog sa:

```
class Student
{
  private:
        //član klase sa imenom studenta (Ante)
        string ime;
        //član klase sa prezimenom studenta (Antić)
        string prezime;
        //član klase sa godinom rođenja studenta (1980,....)
        int god_rodjenja;
        //član klase sa godinom upisa studenta (1999,2000,..)
        int god_upisa;
        //član klase sa trenutnom godinom studija studenta (1,2,3,4)
        int god_studija;
  public:
        //funkcija postavlja vrijednosti svih pet podatkovnih članova
        //tj. imena, prezimena, godine rođenja, godine upisa i godine studija
        void postavi_sve(string varime, string varprezime, int varrodjenje ,int  varupis, int vargodina);
        //funkcija vraća vrijednost člana klase ime
        string dohvati_ime();
        //funkcija vraća vrijednost člana klase prezime
        string dohvati_prezime();
        //funkcija vraća vrijednost člana klase godina rođenja
        int dohvati_god_rodjenja();
        //funkcija vraća vrijednost člana klase godina upisa
        int dohvati_god_upisa();
        //funkcija vraća vrijednost člana klase godina studija
        int dohvati_god_studija();

};
```

Rješenje:

```
class Student
{
private:
      string ime;
      string prezime;
      int god_rodjenja;
      int god_upisa;
      int god_studija;
public:

      void postavi_sve(string varime, string varprezime, int
                                          varrodjenje , int varupis, int vargodina)
      {
            ime = varime;
            prezime = varprezime;
            god_rodjenja = varrodjenje;
            god_upisa = varupis;
            god_studija = vargodina;
      }

      string dohvati_ime()
      {
            return ime;
```

```
	}

	string dohvati_prezime()
	{
		return prezime;
	}

	int dohvati_god_rodjenja()
	{
		return god_rodjenja;
	}

	int dohvati_god_upisa()
	{
		return god_upisa;
	}

	int dohvati_god_studija()
	{
		return god_studija;
	}

};
```

Zadatak 2.

Klasa ***Student*** iz **zadatka 1.** proširena je sa podatkovnim članom *vector<int>sifre_kolegija*. Kontejner *sifre_kolegija* sadrži podatke tipa *int* sa šiframa kolegija koje je student upisao. Kontejner može imati proizvoljan broj elemenata (od 0 do n). Napisati implementaciju novih dodanih metoda (*upisi_sifru, nadi_sifru, dohvati_sve*) za rad sa podatkovnim članom *sifre_kolegija*:

```
class Student
{
  private:
        .....
        .....
        vector<int> sifre_kolegija;
  public:
        .....
        //funkcija upisi_sifru dodaje novu šifru sif u vektor. Pri tome treba voditi //računa da se onemogući unos iste šifre
        dva puta.
        void upisi_sifru(int sif);
        //funkcija nadi_sifru pronalazi šifru sif u vektoru i vraća indeks mjesta na //kojem je šifra u vektoru pronađena, ili
        -1 ako ne nađe šifru
        int nadi_sifru(int sif);
        //funkcija vraća cijeli vektor šifri kolegija
        vector<int> dohvati_sve();
};
```

Rješenje:

```
class Student
{
private:
      string ime;
      string prezime;
      int god_rodjenja;
      int god_upisa;
      int god_studija;
      vector<int> sifre_kolegija;
public:

      void postavi_sve(string varime, string varprezime, int varrodjenje , int
                                                            varupis, int vargodina)
      {
            ime = varime;
            prezime = varprezime;
            god_rodjenja = varrodjenje;
            god_upisa = varupis;
            god_studija = vargodina;
      }
      string dohvati_ime()
      {
            return ime;
      }
      string dohvati_prezime()
      {
            return prezime;
      }
      int dohvati_god_rodjenja()
      {
            return god_rodjenja;
      }
      int dohvati_god_upisa()
      {
```

```
            return god_upisa;
      }
      int dohvati_god_studija()
      {
            return god_studija;
      }
      // funkcija upisi_sifru dodaje novu šifru u vektor. Pri tome treba voditi
      // računa da se onemogući unos iste šifre dva puta.
      void upisi_sifru(int sif)
      {
            for(unsigned int i=0; i < sifre_kolegija.size(); i++)
            {
                  if(sifre_kolegija[i] == sif)
                  {
                        cout << "Unesena sifra vec postoji!\n";
                        return;
                  }
            }
            sifre_kolegija.push_back(sif);
      }
      // funkcija nadi_sifru pronalazi šifru sif u vektoru i vraća index mjesta na
      //kojem je šifra u vektoru pronađena, ili -1 ako ne nađe šifru
      int nadi_sifru(int sif)
      {
            for(unsigned int i=0; i<sifre_kolegija.size(); i++)
            {
                  if(sifre_kolegija[i] == sif)
                  {
                        return i;
                  }
            }
            return -1;
      }
      // funkcija vraća cijeli vektor šifri kolegija
      vector<int> dohvati_sve()
      {
            return sifre_kolegija;
      }

};
```

Zadatak 3.

Napisati definiciju funkcije preopterećenja operatora << za klasu *Student*. koja ispisuje sadržaj objekta u sljedećem formatu:

Ime: Ante
Prezime: Antić
God_rodjenja: 1985
God_upisa: 2004
God_studija: 1
Kolegiji: 20344, 1278, 43288

Ukoliko je neki član objekta prazan ispisuje se:

Ime: Nije uneseno
....
Kolegiji: Nisu uneseni

Funkcija je deklarirana sa:

ostream& operator<<(ostream &var1, Student &var2);

Rješenje:

```
ostream& operator<<(ostream &var1, Student &var2)
{
	if(!strcmp(var2.ime.c_str(),""))
		var1 << "Ime: Nije uneseno" << endl;
	else var1 << "Ime: " << var2.ime.c_str() << endl;

	if(!strcmp(var2.prezime.c_str(),""))
		var1 << "Prezme: Nije uneseno" << endl;
	else var1 << "Prezime: " << var2.prezime.c_str() << endl;

	if(var2.god_rodjenja==0)
		var1 << "Godina rodjenja: Nije uneseno" << endl;
	else var1 << "Godina rodjenja: " << var2.god_rodjenja << endl;

	if(var2.god_upisa==0)
		var1 << "Godina upisa: Nije uneseno" << endl;
	else var1 << "Godina upisa: " << var2.god_upisa << endl;

	if(var2.god_studija==0)
		var1 << "Godina studija: Nije uneseno" << endl;
	else var1 << "Godina studija: " << var2.god_studija << endl;

	cout << "Kolegiji: ";
	if(var2.sifre_kolegija.size()==0)
	{
		var1 << "Nisu uneseni" << endl;
		return var1;
	}
	for(unsigned int i=0; i< var2.sifre_kolegija.size(); i++)
	{
		var1 << var2.sifre_kolegija[i] << ", ";
	}
	var1 << endl;
	return var1;
}
/*	Unutar klase treba dodati deklaraciju ove funkcije kao 'friend' funkcije da
bi iz nje mogli pristupiti privatnim članovima klase.	*/

friend ostream& operator <<(ostream &var1, Student &var2);
```

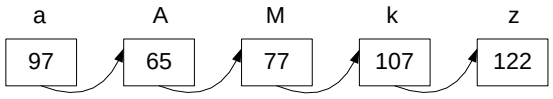
Zadatak 4.

Napisati definiciju funkciju koja vraća listu *int* elemenata. Kao argument funkcija prima string i zatim karaktere string kopira u listu na način da u listu kopira njihove ASCII vrijednosti. Ovo se odnosi samo na mala (a(97) - z(122), A(65) – Z(90)) i velika slova dok se ostali karakteri iz string preskaču. Deklaracija funkcije je:

list*<*int*>** KopirajStringLista(string*** var);

Primjer:

KopirajStringLista("aA!Mk@z") →



Rješenje:

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

	for(unsigned int i=0; i<s.size(); i++)
	{
		int a=(int)s[i];  // sprema ASCII vrijednost znaka u 'a'
		if((a>=65 && a<=90) || (a>=97 && a<=122))   // ako je slovo
			l.push_back(a);
	}
	return l;
}
```

Zadatak 5.

Napisati program kojim se 20 imena *(string)* koje korisnik unosi preko tipkovnice unose u novu kreiranu tekstualnu datoteku. Sadržaj datoteke u svakoj liniji je uneseno ime, zatim razmak (karakter *space*) pa broj slova u unesenom imenu. Podrazumijeva se da korisnik neće pogriješiti prilikom unosa (ne trebaju provjere ispravnosti unosa). Primjer:

korisnikov unos	sadržaj datoteke
Branimir	Branimir 8
Ivan	Ivan 4
Goran	Goran 5

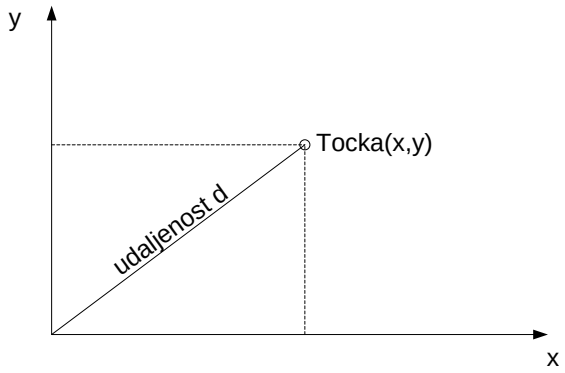
Rješenje:

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

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

	int i=0;
	string s;

	do
	{
		getline(cin,s,'\n');
		file << s << " " << s.size() << endl;
		i++;
	}while(i<20);   // 20 puta

	file.close();

}
```

28.06.2005.

Zadatak 1.

Napisati specifikaciju i implementaciju apstraktnog tipa podatka ***Tocka*** (deklarirati i definirati klasu koja opisuje točku u dvodimenzionalnom koordinatnom sustavu kao na slici). Svi podatkovni članovi klase trebaju biti deklarirani kao **zaštićeni članovi (private)**. Klasa treba imati funkcije za postavljanje i dohvaćanje vrijednosti ***x*** i ***y*** točke. Objekti klase ***Tocka*** trebaju imati integriranu funkcionalnost proračuna udaljenosti od ishodišta koordinatnog sustava (funkcijski član *float udaljenost()* za proračun udaljenosti ***d*** na slici.).



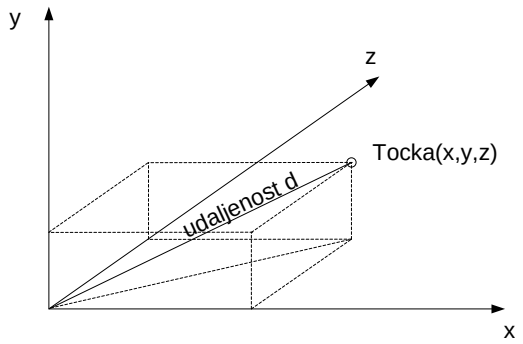
Rješenje:

```
class Tocka
{
private:
	int x;
	int y;

public:
	void set_x(int broj)
	{
		x = broj;
	}

	void set_y(int broj)
	{
		y = broj;
	}

	int get_x()
	{
		return x;
	}

	int get_y()
	{
		return y;
	}

	float udaljenost()
	{
		return sqrt((float)((x*x)+(y*y)));
	}
};
```

Zadatak 2.

Napisati implementaciju apstraktnog tipa podatka ***Tocka3D*** koji opisuje točku u trodimenzionalnom koordinatnom sustavu kao na slici. Klasa nasljeđuje klasu ***Tocka***. Klasa je deklarirana sa:

```
class Tocka3D:public Tocka
{
  private:
        int z;
  public:
        // funkcija postavi_sve postavlja vrijednost podatkovnih članova x, y i z
        void postavi_sve(int vr_x,int vr_y, int vr_z);
        // funkcija vraća vrijednost člana klase z
        int dohvati_z ();
        // nadređena funkcija funkciji udaljenost() iz klase Tocka vraća vrijednost
         // udaljenosti d od ishodišta koordinatnog sustava
        float  udaljenost();
};
```



Rješenje:

```
class Tocka3D:public Tocka
{
private:
      int z;
public:
      void postavi_sve(int vr_x,int vr_y, int vr_z)
      {
            set_x(vr_x);
            set_y(vr_y);
            z = vr_z;
      }

      int dohvati_z()
      {
            return z;
      }

      float  udaljenost()
      {
            int x;
            int y;

            x = get_x();
            y = get_y();

            return sqrt((float)((x*x)+(y*y)+(z*z)));
      }
};
```

Zadatak 3.

Napisati definiciju (kod) funkcije preopterećenja operatora << i >> za unos i za ispis vrijednosti ***x***, ***y*** i ***z*** za klasu ***Tocka3D***. Funkcije su deklarirane sa:

ostream& operator<<(ostream &var1, Tocka3D &var2);

istream& operator>>(istream &var1, Tocka3D &var2);

Rješenje:

```
ostream& operator<<(ostream &var1, Tocka3D &var2)
{
	var1 << "X: " << var2.get_x() << endl;
	var1 << "Y: " << var2.get_y() << endl;
	var1 << "Z: " << var2.z << endl;

	return var1;
}

istream& operator>>(istream &var1, Tocka3D &var2)
{
	int a,b,c;

	if(var1 >> a >> b >> c)
		var2.postavi_sve(a,b,c);

	return var1;
}

/* Unutar klase treba dodati deklaracije ovih dviju funkcija kao 'friend' funkcija
da bi iz njih mogli pristupiti privatnim članovima klase. */

	friend ostream& operator<<(ostream &var1, Tocka3D &var2);
	friend istream& operator>>(istream &var1, Tocka3D &var2);
```

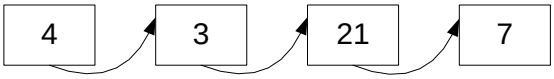
Zadatak 4.

Napisati definiciju (kod) funkcije koja vraća sortirani stog *int* elemenata. Kao argumentfunkcija prima nesortiranu listu *int* elemenata. Deklaracija funkcije je:

stack*<*int*>** SortirajListu(list*<*int*>** var);

Primjer:

lista *var* koja se funkciji predaje kao argument



stog koji se vraća iz funkcije

Rješenje:

```
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;
}
```

Zadatak 5.

Napisati program kojim se tekst koji korisnik unosi preko tipkovnice unosi u novu kreiranu tekstualnu datoteku. Tekst se unosi na način da se sva mala slova koja korisnik upiše preko tipkovnice pretvaraju u velika slova prilikom upisivanja u datoteku. Ostali znakovi se ne mijenjaju.

Primjer:

korisnikov unos	sadržaj datoteke
Danas je utorak 28.6.2005.	DANAS JE UTORAK 28.6.2005.

Rješenje:

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

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

	string s;

	do
	{
		getline(cin,s,'\n');
		for(unsigned int i=0; i<s.size(); i++)
		{
			if(s[i]>='a' && s[i]<='z')  // ako je znak malo slovo
				file << (char)(s[i]-('a'-'A'));
			else if(s[i]>='A' && s[i]<='Z') // ako je znak veliko slovo
				file << (char)(s[i]+('a'-'A'));
			else file << s[i];
		}
		file << endl;
	}while(s.size()!=0);
}

/*

('a'-'A') je brojčana vijednost razlike ASCII vrijednosti malih i velikih slova.
Uvećavajući ili umanjujući ASCII vrijednost znakova za tu razliku zapravo
pretvaramo mala slova u velika i velika u mala. Kasnije, pomoću 'cast' operatora
(char) brojčanu vrijednost vraćamo u char.

*/
```

12.07.2005

Zadatak 1.

Napisati specifikaciju i implementaciju apstraktnog tipa podatka ***kugla*** (deklarirati i definirati klasu koja opisuje kuglu). Svi podatkovni članovi klase trebaju biti deklarirani kao **zaštićeni članovi (private)**. Objekti klase ***kugla*** trebaju imati integriranu funkcionalnost proračuna volumena i oplošja kugle (funkcijske članove za proračun volumena i oplošja kugle).

Rješenje:

```
class Kugla
{
private:
	int radijus;

public:
	Kugla(int r)
	{
		radijus = r;
	}

	void postavi_radijus(int r)
	{
		radijus = r;
	}

	int dohvati_radijus(void)
	{
		return radijus;
	}

	float oplosje(void)
	{
		return (4*radijus*radijus*3.14);
	}

	float volumen(void)
	{
		return ((4/3)*radijus*radijus*radijus*3.14);
	}
};
```

Zadatak 2.

Napisati definiciju funkciju koja vraća vektor *int* elemenata. Kao argument funkcija prima string i zatim karaktere string kopira u vektor na način da u vektor kopira njihove ASCII vrijednosti. Ovo se odnosi samo na mala (a(97) - z(122), A(65) – Z(90)) i velika slova dok se ostali karakteri iz string preskaču. Deklaracija funkcije je:

vektor<int> KopirajStringVektor(***string*** var);

Primjer:

KopirajStringVektor("aA!Mk@z") →

a	A	M	k	z
97	65	77	107	122

Rješenje:

```
vector<int> KopirajStringVektor(string var)
{
	vector<int> v;
	for(unsigned int i=0; i<var.size(); i++)
	{
		// ako je znak veliko ili malo slovo
		if(((var[i]>='a') && (var[i]<='z')) || ((var[i]>='A') &&
								(var[i]<='Z')))
			v.push_back((char)var[i]);
	}
	return v;
}
```

Zadatak 3.

Napisati definiciju (kod) generičke funkcije

template <class T> void OkreniStog(***stack*** <T> &var)

kojom se elementi stoga koji se funkciji predaje po referenci kao argumenti poredaju obrnutim redoslijedom (prvi element ide na posljednje mjesto na stogu, drugi na pretposljednje, ...).

Rješenje:

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

	while(!var.empty())	// dok stog nije prazan
	{
		s.push(var.top());
		var.pop();
	}

	var = s;
}
```

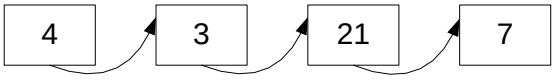
Zadatak 4.

Napisati definiciju (kod) funkcije koja vraća sortirani red *int* elemenata. Kao argument funkcija prima nesortiranu listu *int* elemenata. Deklaracija funkcije je:

queue*<*int*>** SortirajListu(list*<*int*>** var);

Primjer:

lista *var* koja se funkciji predaje kao argument



red koji se vraća iz funkcije

Rješenje:

```
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;
}
```

Zadatak 5.

Napisati program kojim se cijeli sadržaj jedne datoteke kopira u drugu datoteku na način da se na kraju svake kopirane linije doda broj karaktera u originalnoj liniji unutar zagrada kao na primjeru. Svi karakteri se broje (i točka i razmak).

Primjer:

originalna datoteka	kopirana datoteke
Danas je utorak 12.7.2005.	Danas je utorak 12.7.2005. (26)
Ljeto je.	Ljeto je. (9)
Zadnji je ispitni rok.	Zadnji je ispitni rok. (22)

Rješenje:

```
#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());  // dok ne dođe do kraja datoteke

	file1.close();
	file2.close();
}
```

06.09.2005

Zadatak 1.

Napisati implementaciju apstraktnog tipa podatka ***Linija*** koji opisuje liniju u prostoru preko duljine linije. Klasa je deklarirana sa:

```
class Linija
{
  private:
        float duljina;
  public:
        //funkcija postavlja vrijednost duljine
        void set_duljina(float);
        //funkcija vraća vrijednost duljine linije
        float get_duljina();
};
```

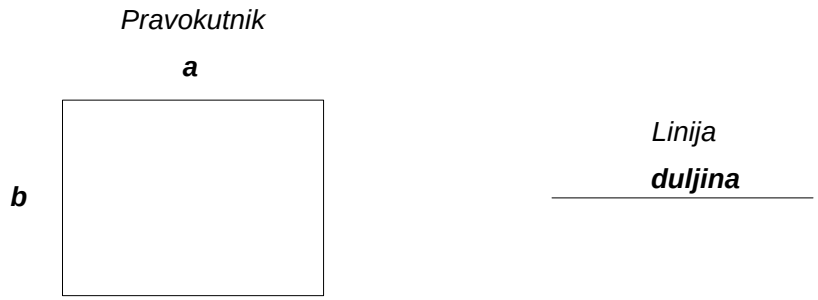
Rješenje:

```
class Linija
{
private:
      float duljina;

public:

      void set_duljina(float d)
      {
            duljina = d;
      }

      float get_duljina()
      {
            return duljina;
      }
};
```

Zadatak 2.

Napisati implementaciju apstraktnog tipa podatka ***Pravokutnik*** koji opisuje pravokutnik preko duljine ***a*** i ***b*** stranice pravokutnika (slika). Klasa nasljeđuje klasu ***Linija*** koja određuje stranicu ***a*** pravokutnika dok je stranica ***b*** definirana kao podatkovni član klase također tipa ***Linija***. Klasa je deklarirana sa:

```
class Pravokutnik:public Linija
{
  private:
        Linija b;
  public:
        //funkcija postavlja duljinu stranice b
        void set_b(float);
        //funkcija vraća vrijednost duljine stranice b
        float get_b ();
        //funkcija vraća površinu pravokutnika
        float get_povrsina();
        //funkcija vraća opseg pravokutnika
        float get_opseg();
};
```



Rješenje:

```
class Pravokutnik:public Linija
{
private:
      Linija b;

public:
      void set_b(float d)
      {
            b.set_duljina(d);
      }

      float get_b()
      {
            return b.get_duljina();
      }

      float get_povrsina()
      {
            return (b.get_duljina()*get_duljina());
      }
      float get_opseg()
      {
            return ((2*b.get_duljina())+(2*get_duljina()));
      }
};
```

Zadatak 3.

Napisati definiciju (kod) funkcije preopterećenja operatora << i >> za klasu *Pravokutnik*. Funkcije su deklarirane sa:

ostream& operator<<(ostream &var1, Pravokutnik &var2);

istream& operator>>(istream &var1, Pravokutnik &var2);

Rješenje:

```
ostream& operator<< (ostream &var1, Pravokutnik &var2)
{
	var1 << "Duljina stranice a: " << var2.get_duljina() << endl;
	var1 << "Duljina stranice b: " << var2.get_b() << endl;

	return var1;
}

istream& operator>> (istream &var1, Pravokutnik &var2)
{
	float d1,d2;

	if(var1 >> d1 >> d2)
	{
		var2.set_duljina(d1);
		var2.set_b(d2);
	}
	return var1;
}

/* Unutar klase treba dodati deklaracije ovih dviju funkcija kao 'friend' funkcija
da bi iz njih mogli pristupiti privatnim članovima klase. */

friend ostream& operator<< (ostream &var1, Pravokutnik &var2);
friend istream& operator>> (istream &var1, Pravokutnik &var2);
```

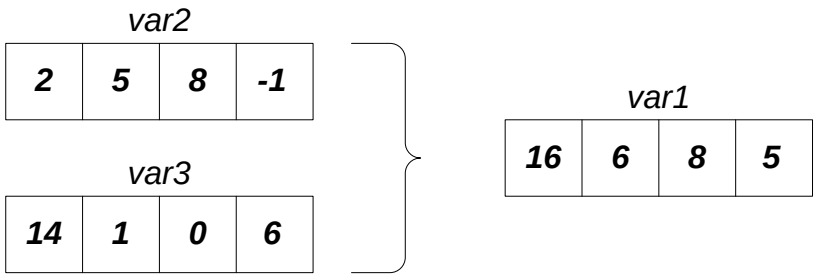
Zadatak 4.

Napisati definiciju (kod) generičke funkcije

template <class T> <T> ZbrojiVektore(***vector*** <T> var2, ***vector*** <T> var3)

kojom se zbrajaju elementi vektora *var2* i vektora *var3* u novi vektor koji se vraća iz funkcije ukoliko vektori *var2* i *var3* imaju isti broj elemenata. Ukoliko vektori *var2* i *var3* nemaju isti broj elemenata funkcija vraća prazni vektor. (Napomena: tip T je numerički)

Primjer:



Rješenje:

```
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;
	}
}
```

Zadatak 5.

Napisati definiciju (kod) funkcije kojom se elementi stoga (sa integer elementima) koji se funkcije predaje kao argument kopiraju u red koji se vraća iz funkcije. Deklaracija funkcije je:

queue<int> KopirajStogRed(***stack<int>*** var1)

Rješenje:

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

	while(!var1.empty())	// dok stog nije prazan
	{
		q.push(var1.top());
		var1.pop();
	}
	return q;
}
```

20.09.2005.

Zadatak 1.

Napisati specifikaciju i implementaciju apstraktnog tipa podatka ***valjak*** (deklarirati i definirati klasu koja opisuje valjak). Svi podatkovni članovi klase trebaju biti deklarirani kao **zaštićeni članovi (private)**. Objekti klase ***valjak*** trebaju imati integriranu funkcionalnost proračuna volumena i oplošja valjka (funkcijske članove za proračun volumena i oplošja valjka).

Rješenje:

```
class Valjak
{
private:
	int radijus;
	int visina;

public:
	Valjak(int r, int v)
	{
		radijus = r;
		visina = v;
	}

	void postavi_radijus(int r)
	{
		radijus = r;
	}

	void postavi_visinu(int v)
	{
		visina = v;
	}

	int dohvati_radijus(void)
	{
		return radijus;
	}

	int dohvati_visinu(void)
	{
		return visina;
	}

	float volumen()
	{
		return radijus*radijus*3.14*visina;
	}

	float oplosje()
	{
		float oplosje_baze;
		float oplosje_plasta;

		oplosje_baze = radijus*radijus*3.14;
		oplosje_plasta = 2*radijus*3.14*visina;

		return ((2*oplosje_baze) + oplosje_plasta);
	}
};
```

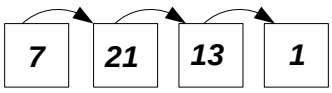
Zadatak 2.

Napisati definiciju funkcije kojom se lista int elemenata duplicira na način da se iza prvog elemenata doda još jednom zadnji element, iza drugog elemenat se još jednom doda predzadnji element, ..., iza zadnjeg elementa se još jednom doda prvi element. Duplicirana lista treba biti pohranjena u originalnoj listi koja se funkciji predaje preko reference. Deklaracija funkcije je:

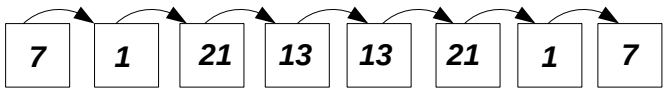
void DuplaLista(***list*** <int> &var1);

Primjer:

originalna lista



lista nakon poziva funkcije DuplaLista



Rješenje:

```
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;
}
```

Zadatak 3.

Napisati definiciju preopterećenja operatora < (manje od) za standardnu klasu *stack*. Deklaracija funkcije je:

bool operator<(***stack*** <int> &prvi, ***stack*** <int> &drugi);

Operator < treba omogućiti uspoređivanje dva stoga na način da ukoliko stog s lijeve strane operatora ima više elemenata taj stog je veći, ukoliko stog s desne strane operatora ima više elemenata taj stog je veći. Ukoliko oba stoga imaju jednaki broj elemenata treba uspoređivati elemente jedan po jedan i onaj stog koji ima više većih elemenata je i ukupno veći.

Rješenje:

```
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  // ako su stogovi jednake veličine
	{
		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;
	}
}
```

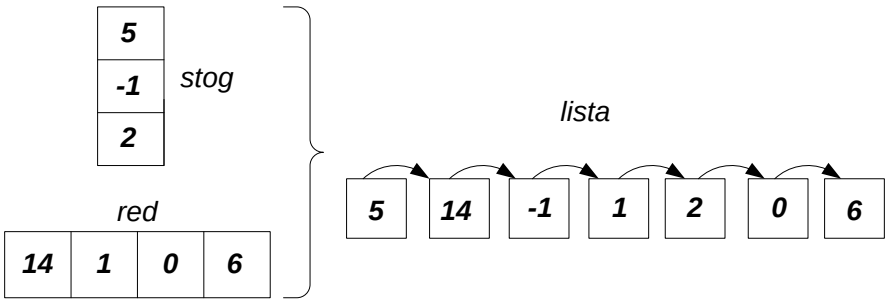
Zadatak 4.

Napisati definiciju (kod) generičke funkcije

template <class T> ***list*** <T> Kopiraj(***stack*** <T> var1, ***queue*** <T> var2)

kojom se elementi stoga *var1* i reda *var2* kopiraju u listu koja se vraća iz funkcije. Kopiranje elemenata stoga i reda se radi naizmjenice tako da se u listu kopira 1. element stoga, zatim 1. element reda, zatim 2. element stoga pa 2. element reda sve dok se ne dodje do kraja stoga i reda. Stog i red mogu imati različiti broj elemenata. U tom slučaju se ostatak stoga ili ostatak reda kopira do kraja u listu.

Primjer:



Rješenje:

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

	while(!(var1.empty() && var2.empty()))  // dok nisu prazni i stog i red
	{
		if(!var1.empty())    // ako stog nije prazan
		{
			l.push_back(var1.top());
			var1.pop();
		}
		if(!var2.empty())    // ako red nije prazan
		{
			l.push_back(var2.front());
			var2.pop();
		}
	}
	return l;
}
```

Zadatak 5.

Napisati program kojim se sadržaj jedne tekstualne datoteke kopira u drugu na način da se sva velika slova kopiraju kao mala slova, a sva mala slova se kopiraju u novu datoteku kao velika slova. Ostali znakovi se kopiraju bez promjene.

Primjer:

originalna datoteka	kopirana datoteka
Danas je utorak 20.9.2005.	dANAS JE UTORAK 20.9.2005.
Ljetu je kraj.	lJETU JE KRAJ.
Počinje novi semestar.	pOČINJE NOVI SEMESTAR.

Rješenje:

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

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

	char ch;

	do
	{
		file1.get(ch);
		if(ch>='a' && ch<='z')  // ako je znak malo slovo
			file2 << (char)(ch - ('a' - 'A'));
		else if(ch>='A' && ch<='Z')   // ako je znak veliko slovo
			file2 << (char)(ch + ('a' - 'A'));
		else   // znak nije slovo
			file2 << ch;

	}while(!file1.eof());  // dok ne dođe do kraja datoteke

	file1.close();
	file2.close();
}


/*

('a'-'A') je brojčana vijednost razlike ASCII vrijednosti malih i velikih slova.
Uvećavajući ili umanjujući ASCII vrijednost znakova za tu razliku zapravo
pretvaramo mala slova u velika i velika u mala. Kasnije, pomoću 'cast' operatora
(char) brojčanu vrijednost vraćamo u char.

*/
```

28.09.2005.

Zadatak 1.

Napisati implementaciju apstraktnog tipa podatka ***Racun***. Klasa je deklarirana sa:

```
class Racun
{
  private:
        string ime_vlasika;
        float stanje_racuna;
        int broj_racuna;
  public:
        //funkcija postavlja vrijednosti sva tri podatkovna člana
        void set_sve(string var1,float var2,int var3);
        //funkcija vraća vrijednost člana klase ime_ vlasika
        string get_ime();
        //funkcija vraća vrijednost člana klase stanje_racuna
        float get_stanje();
        //funkcija vraća vrijednost člana klase broj_racuna
        int get_broj();
};
```

Rješenje:

```
class Racun
{
private:
      string ime_vlasika;
      float stanje_racuna;
      int broj_racuna;

public:
      void set_sve(string var1, float var2, int var3)
      {
            ime_vlasnika = var1;
            stanje_racuna = var2;
            broj_racuna = var3;
      }

      string get_ime()
      {
            return ime_vlasnika;
      }

      float get_stanje()
      {
            return stanje_racuna;
      }

      int get_broj()
      {
            return broj_racuna;
      }
};
```

Zadatak 2.

Napisati implementaciju apstraktnog tipa podatka ***Kredit***. Klasa nasljeđuje klasu ***Racun***. Klasa je deklarirana sa:

```
class Kredit:public Racun
{
  private:
        int kreditna_partija;
        float stanje_kredita;
  public:
//nadređena funkcija set_sve funkciji set_sve iz klase Racun postavlja
//vrijednost podatkovnih članova  ime_vlasika, stanje_racuna, broj_racuna,
//kreditna_partija i stanje_kredita pri čemu stanje_ kredita ne smije biti veće od 10 puta //stanje_racuna. Ukoliko se pokuša postaviti stanje_kredita veće od 10 puta stanje_racuna //tada se stanje_kredita treba postaviti na 0.
        void set_sve (string var1,float  var2,int var3,int var4, float var5);
        //funkcija vraća vrijednost člana klase kreditna_partija
        int get_partija ();
        //funkcija vraća vrijednost člana klase stanje_ kredita
        float get_stanje_kredita();
};
```

Rješenje:

```
class Kredit:public Racun
{
private:
      int kreditna_partija;
      float stanje_kredita;

public:
      void set_sve(string var1, float  var2, int var3, int var4, float var5)
      {
            Racun::set_sve(var1, var2, var3);
            kreditna_partija = var4;
            if(var5 <= (10*get_stanje()))
                  stanje_kredita = var5;
            else
                  stanje_kredita = 0;
      }

      int get_partija()
      {
            return kreditna_partija;
      }

      float get_stanje_kredita()
      {
            return stanje_kredita;
      }
};
```

Zadatak 3.

Napisati definiciju (kod) funkcije preopterećenja operatora << i >> za klasu *Kredit*. Funkcije su deklarirane sa:

ostream& operator<<(ostream &var1, Kredit &var2);

istream& operator>>(istream &var1, Kredit &var2);

Rješenje:

```
ostream& operator<<(ostream &var1, Kredit &var2)
{
	var1 << "Ime vlasnika: " << var2.get_ime() << endl;
	var1 << "Stanje racuna: " << var2.get_stanje() << endl;
	var1 << "Broj racuna: " << var2.get_broj() << endl;
	var1 << "Kreditna paticija: " << var2.kreditna_partija << endl;
	var1 << "Stanje kredita: " << var2.stanje_kredita << endl;

	return var1;
}

istream& operator>>(istream &var1, Kredit &var2)
{
	string i;
	float s;
	int b;
	int k;
	float st;

	if(var1 >> i >> s >> b >> k >> st)
	{
		var2.set_sve(i, s, b, k, st);
	}

	return var1;
}

/* Unutar klase treba dodati deklaracije ovih dviju funkcija kao 'friend' funkcija
da bi iz njih mogli pristupiti privatnim članovima klase. */

friend ostream& operator<<(ostream &var1, Kredit &var2);
friend istream& operator>>(istream &var1, Kredit &var2);
```

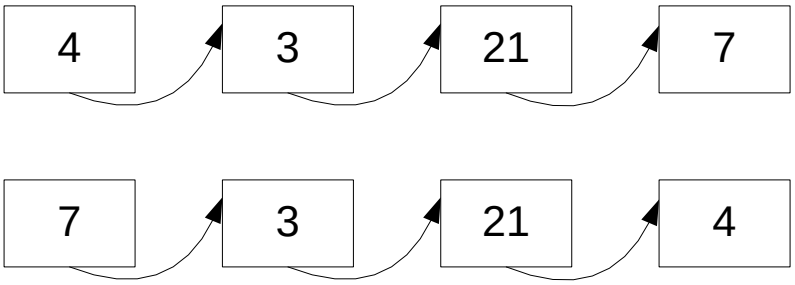
Zadatak 4.

Napisati definiciju (kod) generičke funkcije

template <class T> void ListaPrviZadnji(***list*** <T> &var)

kojom se elementi liste koja se funkciji predaje po referenci kao argumenti poredaju na način da se zamijene samo prvi i posljednji član liste dok svi ostali elementi liste ostaju na svom mjestu.

Primjer:



Rješenje:

```
template <class T> void ListaPrviZadnji(list<T> &var)
{
	T t1,t2;

	t1 = var.front();
	t2 = var.back();

	var.pop_front();
	var.pop_back();

	var.push_back(t1);
	var.push_front(t2);

}
```

Zadatak 5.

Napisati program kojim se tekst koji korisnik unosi preko tipkovnice unosi u novu kreiranu tekstualnu datoteku. Tekst se unosi na način da se u datoteku upisuju samo karakteri koja korisnik upiše preko tipkovnice dok se brojevi ne upisuju u datoteku. Unos završava kada korisnik upiše prazni string preko tipkovnice.

Primjer:

korisnikov unos	sadržaj datoteke
Danas je srijeda 28.9.2005.	Danas je srijeda ...
Dekanski je rok.	Dekanski je rok.
Nastava počinje 3.10.	Nastava počinje ..

Rješenje:

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

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

	string s;
	char c;

	do
	{
		getline(cin,s,'\n');
		for(int i=0; i<s.size(); i++)
		{
			if(s[i]<'0' || s[i]>'9')    // ako nije broj
				file << s[i];
		}
		file << endl;
	}while(s.size()!=0);  // dok se ne unese prazan string

	file.close();
}
```