Kolokvijum iz ORS, maj 2005.

Ideja zadatka: U magacinu pravougaonog oblika nalaze se paketi. Svaki paket nosi po jedan nosač, koji pokušava da iznese paket iz magacina. Paketi su pravougaonog oblika. Smatrati da se prenošenje izvodi isključivo transliranjem. Zadatak je simulirati iznošenje paketa iz magacina ako je poznata početna konfiguracija, koju čine položaji paketa u magacinu i poznato ponašanje nosača. Radi pojednostavljenja, pakete i nosače ćemo posmarati kao celinu koju ćemo označavati imenom Paket.

1. deo (10 poena): Napisati klasu *Paket* koja predstavlja baznu klasu za sve konkretne vrste paketa. Svaki paket ima položaj (celobrojne koordinate x i y), celobrojnu širinu i dužinu i oznaku kojom se predstavlja. Paketi se pomeraju primenom metoda:

```
bool Pomeri( Magacin& m )
```

koji vraća true akko je paket pomeren, kojom prilikom menja položaj paketa za najviše jedno jedinično mesto.

2. deo (20 poena): Napisati klasu *Magacin*. Magacin sadrži pakete koji u u njemu. Vrata se nalaze na prednjem zidu magacina, imaju poznat položaj i uvek su otvorena. Napisati bar metode:

```
Magacin( int širina, int visina, int počVrata, int krajVrata )
```

`bool SlobodanProstor( const Paket& p ) const` – proverava da li je slobodan prostor na koji bi trebalo da se postavi paket

`bool DodajPaket( const Paket& p )` – ako je odgovarajući prostor slobodan, dodaje kopiju paketa u magacin, i vraća true, inače vraća false

Napisati i sve ostale metode koji su neophodni za implementaciju navedenih metoda. Sugestija: voditi nezavisno matricu polja, kojom se modelira raspored paketa u magacinu (paket zauzima onoliko polja kolike su mu dimenzije) i kolekciju paketa koji su u magacinu.

3. deo (20 poena): Napisati metod klase Magacin koji simuliraju po jedan korak kretanja svakog paketa:

```
bool JedanPotez()
```

koji vraća true akko je bar jedan paket pomeren ili iznešen iz magacina. Smatrati da se paketi pomeraju jedan po jedan, a u jednom potezu je potrebno svim paketima pružiti priliku da se pomere. Izbor novog položaja izvodi sam objekat paket. Pakete koji siđu u liniju vrata (tj. bar jednim svojim delom se preklope sa vratima) smatrati za iznešene i uklanjati iz magacina. Napisati i sve ostale metode koji su neophodni za implementaciju ovog metoda (osim onih koji se odnose na 2. deo).

4. deo (15 poena): Napisati klasu *UporanPaketLevo*. Radi se o paketu čiji nosači nastavljaju da se kreću u istom smeru sve dok ne udare u zid ili u drugi paket. U tom slučaju pokušavaju da promene smer kretanja ulevo za 90, 180 ili 270 stepeni, a što je manje moguće.

5. deo (15 poena): Napisati klasu *NervozanPaket*. Radi se o paketu čiji nosači teže da menjaju smer kretanja nakon datog broja koraka (parametar konstruktora). Koraci se broje od poslednje promene smera. Ako se pre željenog broja koraka udari u nešto, menja se smer. Promena smera se izvodi, ako je moguće, za 90 stepeni ulevo ili udesno, a ako to nije moguće, onda za 180 stepeni.

6. deo (10 poena): Podržati ispisivanje stanja magacina u tok, u obliku navedenom u primeru.

7. deo (10 poena): Napisati program koji postavlja neku početnu konfiguraciju magacina (eksplicitno, bez čitanja datoteke, pravi magacin i postavlja nekoliko paketa) i izvodi korake simulacije sve dok se iz magacina ne iznesu svi paketi ili raspored paketa ne bude takav da je dalje pomeranje nemoguće.

Primer konfiguracije magacina:

```
        ...
        Magacin g(20,10,3,5); // sirina 20, visina 10, vrata na poljima 3 do 5 (ukljucujuci)
        g.DodajPaket( UporanPaketLevo('a',2,1,3,2)); // oznaka "a", polozaj (2,1), dimenzije (3,2)
        g.DodajPaket( UporanPaketLevo('b',4,6,1,3));
        g.DodajPaket( NervozanPaket('c',6,8,3,2,2)); // menja smer nakon 2 koraka
        g.DodajPaket( NervozanPaket('d',14,5,2,3,4)); // menja smer nakon 4 koraka
        ...
```

zapisuje se u tok kao:

```
======================
|                    |
|  aaa               |
|  aaa               |
|                    |
|                    |
|              dd    |
|    b         dd    |
|    b         dd    |
|    b ccc           |
|      ccc           |
====   ===============
```

Kolokvijum se rešava 3 sata. Za pozitivnu ocenu je neophodno 50 poena. Delovi zadatka se ne moraju rešavati redom. Pisati čitko i sa potrebnim komentarima. U svim delovima zadatka je dopušteno (i preporučljivo) upotrebljavati standardnu biblioteku.

Rešenje

```
#include <iostream>
#include <vector>

using namespace std;

class Magacin;

class Paket
{
public:
    Paket( char sb, int x, int y, int s, int d, int sm )
        : xpos(x), ypos(y), sirina(s), duzina(d),
          smer(sm), simbol(sb)
        {}
    virtual ~Paket() {}
    virtual Paket* Kopija() const = 0;

    bool Pomeri( Magacin& g );
    virtual void prioritetSmerova( int smerovi[] ) const = 0;

    int dx() const
        { return dx(smer); }
    int dy() const
        { return dy(smer); }

    int xpos, ypos;
    int sirina, duzina;
    int smer;
    char simbol;

private:
    enum { sLevo, sGore, sDesno, sDole };
    static int dx( int s )
        {
        static int _dx[] = { -1, 0, 1, 0 };
        return _dx[s];
        }
    static int dy( int s )
        {
        static int _dy[] = { 0, -1, 0, 1 };
        return _dy[s];
        }
};

class Magacin
{
public:
    Magacin( unsigned s, unsigned v, unsigned vPoc, unsigned vKraj )
        { postaviVelicinu( s, v, vPoc, vKraj ); }
    ~Magacin()
        { deinit(); }
    Magacin( const Magacin& g )
        { init(g); }
    Magacin& operator = ( const Magacin& g )
        {
        if( this != &g ){
            deinit();
            init(g);
            }
        return *this;
        }

    bool SlobodanProstor( int x, int y, int s, int d ) const
        {
        for( int i=0; i<s; i++ )
            for( int j=0; j<d; j++ )
                if( !slobodnoPolje( x + i, y + j ) )
                    return false;
        return true;
        }
    bool SlobodanProstor( const Paket& p ) const
        { return SlobodanProstor( p.xpos, p.ypos, p.sirina, p.duzina ); }

    bool DodajPaket( const Paket& p )
        {
        if( !SlobodanProstor( p ))
            return false;
        _paketi.push_back( p.Kopija() );
        postaviPaketNaPolja( p );
        return true;
        }
```

```
    bool JedanPotez()
        {
        bool biloPomeranja = false;
        for( unsigned i=0; i<_paketi.size(); i++ ){
            obrisiPaketSaPolja( *_paketi[i] );
            bool pomeren = _paketi[i]->Pomeri( *this );
            if( pomeren && izasao( *_paketi[i] )){
                delete _paketi[i];
                _paketi.erase( _paketi.begin() + i );
                i--;
                }
            else
                postaviPaketNaPolja( *_paketi[i] );
            biloPomeranja = biloPomeranja | pomeren;
            }
        return biloPomeranja;
        }

    void Ispis( ostream& ostr ) const
        {
        for( int i=-1; i<=_sirina; i++ )
            ostr << '=';
        ostr << endl;
        for( int i=0; i<_duzina; i++ ){
            ostr << '|';
            for( int j=0; j<_sirina; j++ )
                ostr << _polja[j][i];
            ostr << '|' << endl;
            }
        for( int i=-1; i<=_sirina; i++ )
            ostr << ( i>=_vrataPocetak && i<=_vrataKraj ? ' ' : '=' );
        ostr << endl;
        }

private:
    void postaviPaketNaPolja( const Paket& p )
        { oznaciPolja( p.xpos, p.ypos, p.sirina, p.duzina, p.simbol ); }
    void obrisiPaketSaPolja( Paket& p )
        { oznaciPolja( p.xpos, p.ypos, p.sirina, p.duzina, ' ' ); }
    void oznaciPolja( int x, int y, int s, int v, char sb )
        {
        for( int i=0; i<s; i++ )
            for( int j=0; j<v; j++ )
                _polja[x+i][y+j] = sb;
        }

    bool slobodnoPolje( int x, int y ) const
        {
        return  ( x>=0 && x<_sirina && y>=0 && y<_duzina
                  && _polja[x][y] == ' ')
                ||
                ( y==_duzina && x>=_vrataPocetak && x<=_vrataKraj);
        }

    bool izasao( const Paket& p ) const
        {
        return p.ypos + p.duzina > _duzina
            && p.xpos >= _vrataPocetak
            && p.xpos + p.sirina - 1 <= _vrataKraj;
        }

    void postaviVelicinu( int s, int d, int vPoc, int vKraj )
        {
        _sirina = s;
        _duzina = d;
        _vrataPocetak = vPoc;
        _vrataKraj = vKraj;
        _paketi.clear();
        _polja.resize( _sirina );
        for( int i=0; i<_sirina; i++ ){
            _polja[i].resize( _duzina );
            for( int j=0; j<_duzina; j++ )
                _polja[i][j] = ' ';
            }
        }
    void init( const Magacin& g )
        {
        postaviVelicinu( g._sirina, g._duzina, g._vrataPocetak, g._vrataKraj );
        for( unsigned i=0; i<g._paketi.size(); i++ )
            DodajPaket( *g._paketi[i] );
        }
    void deinit()
        {
        for( unsigned i=0; i<_paketi.size(); i++ )
            delete _paketi[i];
        }

    int _sirina, _duzina;
    int _vrataPocetak, _vrataKraj;
    vector< Paket* > _paketi;
    vector< vector<char> > _polja;
```

```
};
ostream& operator << ( ostream& ostr, const Magacin& g )
{
    g.Ispis( ostr );
    return ostr;
}

bool Paket::Pomeri( Magacin& g )
{
    int smerovi[4];
    prioritetSmerova( smerovi );
    for( int i=0; i<4; i++ ){
        int s = smerovi[i];
        int dx = this->dx(s);
        int dy = this->dy(s);
        if( g.SlobodanProstor( xpos+dx, ypos+dy, sirina, duzina )){
            xpos += dx;
            ypos += dy;
            smer = s;
            return true;
            }
        }
    return false;
}

class UporanPaketLevo : public Paket
{
public:
    UporanPaketLevo( char sb, int x, int y, int s, int d )
        : Paket( sb, x, y, s, d, random(4) )
        {}

    virtual Paket* Kopija() const
        { return new UporanPaketLevo(*this); }

    void prioritetSmerova( int smerovi[] ) const
        {
        for( int i=0; i<4; i++ )
            smerovi[i] = ( smer + 4 - i )%4;
        }
};

class NervozanPaket : public Paket
{
public:
    NervozanPaket( char sb, int x, int y, int s, int d, int str )
        : Paket( sb, x, y, s, d, random(4) ),
          _strpljenje(str), _brojkoraka(0)
        {}

    virtual Paket* Kopija() const
        { return new NervozanPaket(*this); }

    void prioritetSmerova( int smerovi[] ) const
        {
        int i = 0;
        if( _brojkoraka )
            smerovi[i++] = smer;
        int p = (smer + 1 + 2*random(2) ) % 4;
        smerovi[i++] = p;
        smerovi[i++] = ( p + 2 ) % 4;
        smerovi[i++] = (smer + 2) % 4;
        if( !_brojkoraka )
            smerovi[i++] = smer;
        _brojkoraka = ( _brojkoraka + 1 ) % _strpljenje;
        }

private:
    int _strpljenje;
    mutable int _brojkoraka;
};


main()
{
    Magacin g(20,10,3,5);
    g.DodajPaket( UporanPaketLevo('a',2,1,3,2));
    g.DodajPaket( UporanPaketLevo('b',4,6,1,3));
    g.DodajPaket( NervozanPaket('c',6,8,3,2,2));
    g.DodajPaket( NervozanPaket('d',14,5,2,3,4));
    cout << g << endl;

    while( cin.get() ){
        if( !g.JedanPotez() )
            break;
        cout << endl << endl << g << endl;
        }
}
```