KLASIŲ HIERARCHIJOS KŪRIMO PROGRAMOS PAVYZDYS

Tiesinės duomenų struktūros: sąrašas, stekas, eilė.

Programos pavyzdys iliustruoja dviejų aptarnavimo disciplinų rišlaus sąrašo kūrimą. Pats rišlus sąrašas apibrėžtas kaip abstrakti klasė, o stekas ir eilė realizuojami klasių hierarchijoje, sukurtoje abstrakčios klasės pagrindu. Stekas ir eilė turi skirtingas realizacijas, tačiau prieigai prie jų naudojama vienoda sąsaja (interfeisas).

Pavyzdys P2. Dinaminio polimorfizmo taikymas kuriant klasių hierarchiją.

```
#include <iostream.h>
#include <stdlib.h>
#include <ctype.h>

class List
{
        int Num;
   public:
        List () { Head = Tail =Next = 0; }
        virtual void Put(int n) = 0;
        virtual bool Get(int& n) = 0;
                void SetValue(int n) { Num= n; };
                int GetValue() { return Num; };
        //Rodyklės
        List* Head; // Rodyklė į sąrašo pradžią
        List* Tail; // Rodyklė į sąrašo pabaigą
        List* Next; // Rodyklė į tolesnį sąrašo elementą
};

class Queue : public List
{
        public:
          void Put(int n);
          bool Get(int& n);
};
void Queue :: Put(int n)
{
    List* Item;
    Item = new Queue;                   // Sukurimas naujas eilės elementas
    Item -> SetValue(n);                // Jam priskiriama reikšmė
    if  (Tail)  Tail->Next = Item;      // Pridedame elementą į eilės galą
    Tail = Item;
```

```
    Item ->Next = NULL;
    if (!Head) Head = Tail;
}
bool Queue :: Get(int& n)
{
    List* Item;
    if (!Head) {
                    n = 0;
                    return false;
                    }
    n = Head-> GetValue();      // Gauname pirmo eilės elemento reikšmę
    Item = Head;                // Nustatome eilės pradžią į tolesnį elementą
    Head = Head->Next;
    delete Item;                // Šaliname pirmą eilės elementą
    return true;
}

class Stack : public List
{
        public:
          void Put(int n);
          bool Get(int& n);
};
void Stack :: Put(int n)
{
    List* Item;
    Item = new Stack;                   // Sukurimas naujas steko elementas
    Item -> SetValue(n);                // Jam priskiriama reikšmė
    if  (Head)  Item->Next = Head;      // Pridedame elementą į steką
    Head = Item;
    if (!Tail) Tail = Head;
}
bool Stack :: Get(int& n)
{
    List* Item;
    if (!Head) {
                    n = 0;
                    return false;
                    }
    n = Head-> GetValue();      // Gauname pirmo steko elemento reikšmę
    Item = Head;                // Nustatome steko pradžią į tolesnį elementą
    Head = Head->Next;
    delete Item;                // Šaliname pirmą steko elementą
    return true;
}
```

```
int main ()
{
        List* Item;
        int saved;
        char symbol;
        Queue QueOb;
        Stack StOb;

        for (int i=0; i<10; i++)
        {
         cout << “Į kurią struktūrą talpinti reikšmę: į eilę ar į steką? (E/S):  ” ;
         cin >> symbol;
         symbol = tolower(symbol);
         if (symbol == ‘e’) Item = &QueOb;
         else if (symbol == ‘s’) Item = &StOb;
         else break;
         Item ->Put(i);
        }

        cout << “Iš kurios struktūros gauti reikšmę: iš eilės ar steko? (E/S): ” <<endl”
         << “Norėdami baigti darbą įveskite B “<<endl;
        for (; ;)
        {
         cin >> symbol;
         symbol = tolower(symbol);
         if (symbol == ‘b’) break;
         if (symbol == ‘e’) Item = &QueOb;
         else if (symbol == ‘s’) Item = &StOb;
         else break;
         if (Item->Get(saved))
                        cout << saved << endl;
         else cout << “Sąrašas tuščias” << endl;
        }
        return 0;
    }
```