ABSTRAKTNE KLASS

Kui virtuaalne meetod on võrdsustatud nulliga

virtual int meetod(char *) = 0;

siis nimetatakse seda funktsiooni ehtsaks virtuaalfunktsiooniks (*pure virtual function*).

Tuletatud klassis on kohustus kirjeldada ehtne virtuaalfunktsioon kas:

- "tegeliku" meetodina

või

- jälle ehtsa virtuaalfunktsioonina

Klassi, milles on vähemalt üks ehtne virtuaalfunktsioon, nimetatakse abstraktseks klassiks:

- ei saa deklareerida "abstraktset" objekti, s.t. abstraktne klass on mõeldud ainult tuletamiseks;

- küll aga on võimalik viitmuutuja abstraktsele klassile

Näide:

Nimistud, milles on segamini stringid ja täisarvud

```
#include <iostream.h>
#define  NULL  0
#define  NL  '\n'

struct  Item
{
        void  *data;
        Item  *next;
        Item( void ) { next = NULL; }
        virtual void printItem( void ) = 0;      // ehtne virtuaalne!
}

class StrItem : Item
{
public:
        StrItem( char *string = NULL )  { data = string; }
        void printItem( void )
        {
                cout << (char *) data << NL;
        }
}

class IntItem : Item
{
public:

        IntItem( int *p = NULL ) { data = p }
        void printItem( void )
        {
                cout << * (int *) data << NL;
        }
}
```

```
class List
{
protected:
        Item *top;
        List( void )  { top = NULL;  }
        virtual void addItem( void * ) = 0;      // ehtne virtuaalne!
public:
        void printList (void )
        {
                Item *jooksev = top;
                while( jooksev )
                {
                        jooksev -> printItem( );  // NB! hiline sidumine!
                        jooksev = jooksev -> next;
                };
        }
}

class LiFo : public List
{
public:

        // vaikimisi on konstruktoriks  LiFo( void ) : List( void )  {  }

        void addItem( Item *uus )
        {
                Item *eelmine = top;
                top = uus;
                top -> next = eelmine;
        }
}
```

```
class FiFo : public List
{
	Item *last;
public:
	FiFo( void )  { last = NULL;  }
	void addItem( Item *uus )
	{
		Item *viimane = last;
		if( ! top ) top = uus;
		last = uus;
		if( viimane ) viimane -> next = uus;
	}
}



// testprogramm

int main( void )
{
	FiFo  saba;
	StrItem  k( “Kaarel” ),  j( “Jüri” );
	IntItem   a( 1945 ),  b( 1949 );

	saba.addItem(&k );
	saba.addItem(&b );
	saba.addItem(&a );
	saba.addItem(&j );
	saba.printList( );

	return 0;
}
```