Лабораторная работа № 4

ТЕМА: **Принципы наследования и полиморфизма на C++. Организация шаблонов на языке C++. Библиотека STL.**

ЦЕЛЬ: Изучение принципов разработки объектно-ориентированных программ со схемами наследования. Создание класса по обработке данных списка, реализующего принципы полиморфизма.

1 Теоретические вопросы подготовки к работе:

1. Организация наследования в C++.
2. Схемы взаимодействия классов.
3. Организация шаблонов в C++.
4. Шаблонные функции и шаблонные классы.
5. Полиморфизм.
6. Реализация списочных структур в C++.
7. Библиотека Standard Template Library.

2 Контрольные вопросы по теме:

1. Дайте определение наследованию.
2. Какими иерархическими схемами может быть описано взаимодействие объектов?
3. Приведите описание типов и видов наследования.
4. Приведите формальное определение понятий шаблон функции и шаблон класса.
5. Понятие полиморфизма.
6. Особенности использования виртуальных функций.
7. Отличие «позднего» связывания от «раннего».
8. Назначение виртуального деструктора.
9. Как полиморфизм способствует расширяемости?
10. Выявите различия между наследованием интерфейса и реализаций. Чем отличаются иерархические структуры, разработанные для наследования интерфейса, от иерархических структур, разработанных для наследования реализаций?
11. Понятие абстрактного класса.
12. Особенности реализации списочных структур в C++. Почему для формирования перечисленных структур используется динамическая память?
13. Назначение стандартной библиотеки по работе со сложными структурами данных (STL).
14. Приведите описание однонаправленного, двунаправленного списка, дека, стека (LIFO), очереди (FIFO) на базе абстрактных классов однонаправленного и двунаправленного списка на C++.
15. Логику какого оператора языка C++ реализуют в полиморфном программировании виртуальные функции и динамическое связывание?
16. Какие требования предъявляются к переопределенной виртуальной функции в производном классе?

Примечание. Контрольный вопрос № 14 для группы А выполняется программно, для приема предоставляются листинги.

3 Индивидуальные задания:

3.1 Основные задания:

Используя стандартную библиотеку шаблонов STL и класс list по работе с двунаправленным списком, выполнить следующие задачи:

1) Даны натуральное число n, действительные числа $a_1, a_2, ..., a_n$. Если в последовательности $a_1, a_2, ..., a_n$ есть хотя бы один элемент, меньший, чем (-3), то все отрицательные элементы заменить их квадратами, оставив остальные элементы без изменения; в противном случае умножить все элементы на 0.1. Результирующий список вывести в порядке $a_n, a_{n-1}, ..., a_1$.

2) Даны натуральное число n, действительные числа $a_1, a_2, ..., a_n$. Если последовательность $a_1, a_2, ..., a_n$ упорядочена по неубыванию, то оставить ее без изменения. Иначе получить последовательность $a_n, a_{n-1}, ..., a_1$. Результирующий список вывести в порядке $a_1, a_2, ..., a_n$.

3) Даны натуральное число n, действительные числа $x_1, x_2, ..., x_n$. Вычислить:

А) $x_1 x_n + x_2 x_{n-1} + ... + x_n x_1$

Б) $(x_1 + x_n)(x_2 + x_{n-1})...(x_n + x_1)$

4) «Считалка». Даны натуральные n, m. Предполагается, что n человек встают в круг и получают номера, считая против часовой стрелки, $1,2,..., n$. Затем, начиная с первого, также против часовой стрелки отсчитывается m-й человек (поскольку люди стоят по кругу, то за n-м человеком стоит первый). Этот человек выходит из круга, после чего, начиная со следующего, снова отсчитывается m-й человек и так до тех пор, пока из всего круга не останется один человек. Определить его номер.

5) Даны натуральные числа n, m, символы $s_1, s_2, ..., s_n, (m < n)$. Получить последовательность символов:

А) $s_{m+1}, s_{m+2}, ..., s_n, s_1, ..., s_m$

Б) $s_{m+1,} s_{m+2}, ..., s_n, s_m, ..., s_1$

6) Даны натуральное число n, целые числа $a_1, a_2, ..., a_{2n}$. Определить истинность выполнения для $i = 1, ..., n$:

А) $a_i = 2a_{n-i+1} + a_{2n-i+1}$

Б) $a_i + a_{2n-i+1} > 17$

7) Даны натуральное число n, действительные числа $a_1, a_2, ..., a_n$. Преобразовать данную последовательность, расположив вначале отрицательные члены, а затем - неотрицательные. При этом:

А) порядок отрицательных чисел изменяется на обратный, а порядок неотрицательных сохраняется прежним;

Б) порядок отрицательных чисел сохраняется прежним, а порядок неотрицательных изменяется на обратный.

8) Даны натуральное число n, символы $s_1, s_2, ..., s_n$. Будем рассматривать слова, образованные входящими в последовательность символами. Если общее количество слов больше единицы и нечетно, то удалить первое слово. Затем последовательность должна быть отредактирована следующим образом. Должны

быть удалены группы пробелов, которыми начинается и заканчивается последовательность, а каждая внутренняя группа пробелов должна быть заменена одним пробелом.

9) Даны натуральные числа *k, m, n*, символы $s_1, s_2, ..., s_k, t_1, t_2, ..., t_m, u_1, u_2, ..., u_n$. Получить по одному разу те символы, которые входят одновременно во все три последовательности.

10) Даны натуральное число *n*, целые числа $a_1, a_2, ..., a_n$. Выяснить, имеются ли среди чисел совпадающие.

11) Даны натуральное число *n*, целые числа $a_1, a_2, ..., a_{3n}$. Выяснить наличие для всех $a_{2n+1}, a_{2n+2}, ..., a_{3n}$ равных среди $a_1, a_2, ..., a_{2n}$.

12) Даны натуральное число *n*, действительные числа $r_1, r_2, ..., r_n$. Получить:

А) $r_1, r_2, ..., r_n, r_n, r_{n-1}, ..., r_1$

Б) $r_1, r_2, ..., r_n, r_1, r_2, ..., r_n$

13) Даны натуральное число *n*, действительные числа $a_1, ..., a_{2n}$. Получить:

А) $(a_1 - a_{2n})(a_3 - a_{2n-2})(a_5 - a_{2n-4})...(a_{2n-1} - a_2)$

Б) $a_1 a_{2n} + a_2 a_{2n-1} + ... + a_n a_{n+1}$

14) Даны натуральное число *n*, действительные числа $x_1, x_2, ..., x_n$. Получить:

А) $(x_1 + x_n)(x_2 + x_{n-1})...(x_n + x_1)$

Б) $(x_1 + x_2 + 2x_n)(x_2 + x_3 + 2x_{n-1})...(x_{n-1} + x_n + 2x_2)$

15) Даны натуральное число *n*, действительные числа $a_1, ..., a_{2n}$. Получить:

А) $\min(a_1 + a_{n+1}, a_2 + a_{n+2}, ..., a_n + a_{2n})$

Б) $\max(\min(a_1, a_{2n}), \min(a_2, a_{2n-1}), ..., \min(a_n, a_{n+1}))$

3.2 Усложненные задания:

Для задач основного задания самостоятельно разработать пользовательский базовый класс двунаправленного списка. Определить необходимые методы обработки (создание списка, вставка и удаление определенных элементов, удаление списка). Проект должен содержать базовый класс списка и производный класс, в котором определены методы (поиска, сравнения, изменения значения или изменения положения элементов списка) по правилам в соответствии с вариантом.

Приложение А

Пример программы создания однонаправленного списка

Программа создает однонаправленный список на базе шаблона, заполняет элементы сгенерированными псевдослучайными числами, выводит значения элементов списка и предлагает ввести значение элемента из списка для подсчета количества его вхождений в список.

```
//List.cpp
#include <iostream.h>
#pragma hdrstop
```

```
#include "node.h"
#include "nodelib.h"
#include "random.h"

void main(void)
{                                       //установить голову списка в NULL
        Node<int> *head=NULL, *currPtr;
        int i, key, count=0;
        RandomNumber rnd;       //ввести 10 случайных чисел в начало списка
        for(i=0; i<10; i++)
                InsertFront(head, int(1+rnd.Random(10)));
        cout<<"Spisok:";                //печать исходного списка
        PrintList(head,noNewline);
        cout<<endl;
        cout<<"Input key:";             //запросить ввод ключа
        cin>>key;
        currPtr=head;                    //цикл по списку
        while(currPtr !=NULL)
        {                               //если данные совпадают с ключом, увеличить
count
                if(currPtr->data==key)      count++;
                currPtr=currPtr->NextNode();//перейти к следующему узлу
        }
        cout<<"Value  "<<key<<"  appears "<<count
                <<" once in this list"<<endl;
}

//Node.h
template <class T>
class Node
{       private:
                Node<T> *next;                  //адрес след. узла
        public:
                T data;
                Node (const T& item, Node<T>* ptrnext=NULL); //Конструктор
                void InsertAfter(Node<T> *p); //Методы обработки списка
                Node<T> *DeleteAfter(void);
                Node<T> *NextNode(void) const; //Получить адрес след. узла
};
template <class T>
Node<T>::Node(const T& item, Node<T> *ptrnext) : //Конструктор
                        data(item), next(ptrnext)
{}
                                        //Возвратить закрытый член next
template <class T>
Node<T> *Node<T>::NextNode(void) const
{
        return next;
```

```
}
                                        //Вставить узел p после текущего узла
template <class T>
void Node<T>::InsertAfter(Node<T> *p)
{                                       //p указывает на следующий за текущим узел,
                                        //      а текущий на p
        p->next = next;
        next=p;
}
//Удалить узел, следующий за текущим и возвратить его адрес
template <class T>
Node<T> *Node<T>::DeleteAfter(void)
{                                       //сохранить адрес удаляемого узла
        Node<T> *tempPtr=next;
                                        //если нет след. узла, возвратить NULL
        if(next==NULL)
                return NULL;
                                        //тек. узел указывает на узел, следующий за tempPtr
        next=tempPtr->next;
        return tempPtr;                 //возвратить указатель на несвязанный узел
}
//Nodelib.h
#include <stdlib.h>
//Флажки печати в одну строку или с переводом на новую строку
enum AppendNewline {noNewline, addNewline};
                                        //Выделение узла с данным-членом item
                                        // и указателем nextPtr
template <class T>
Node<T>* GetNode(const T& item, Node<T> *nextPtr=NULL)
{       Node<T> *newNode;
                                        //выделение памяти при передаче item и nextPtr
                                        //конструктору, завершение программы, если
                                        //выделение памяти ошибочно
        newNode=new Node<T>(item,nextPtr);
        if (newNode==NULL)
        {       cerr<<"Memory error"<<endl;
                exit(1);
        }
        return newNode;
}
//Вставка элемента в начало списка
template <class T>
void InsertFront(Node<T>* & head, T item)
{                                       //создание нового узла, чтобы он указывал
                                        //на начальную голову списка, изменение
                                        //головы списка
        head=GetNode(item,head);
}
```

```
//Печать связанного списка
template <class T>
void PrintList(Node<T>*  head, AppendNewline addnl=noNewline)
{                                       //currPtr проходит по списку, начиная с головы
        Node<T>* currPtr=head;
                                        //пока не конец списка, печатать значения
                                        //данных текущего узла
        while(currPtr != NULL)
        {
                if(addnl==addNewline)
                        cout<<currPtr->data<<endl;
                else
                        cout<<currPtr->data<<"  ";
                                        //перейти к следующему узлу
                currPtr=currPtr->NextNode();
        }
}
//Random.h
#include <time.h>
//генерация псевдослучайного числа по текущему базовому значению (seed)
const unsigned long maxshort=65536;
const unsigned long multiplier=1194211693L;
const unsigned long adder=12345L;

class RandomNumber
{
        private:
                                        //текущее seed-значение
                unsigned long randSeed;
        public:
                                        //конструктор; параметр по умолчанию (0)
                                        //задает выбор seed-значения
                RandomNumber(unsigned long s=0);
                                        //генерировать целое число в диапазоне [0,n-1]
                unsigned short Random(unsigned long n);
                                        //генерировать действительное число
                                        //в диапазоне [0,1.0]
                double fRandom(void);

};
                                        //генерация seed-значения
RandomNumber::RandomNumber(unsigned long s)
{
        if (s==0)
                //
                randSeed=time(0);
        else
                //
```

```
        randSeed=s;
}
                                        //вернуть целое 0<=value<=n-1<65535
unsigned short RandomNumber::Random(unsigned long n)
{
                                        //используются константы
    randSeed=multiplier * randSeed * adder;
    return (unsigned short) ((randSeed>>16)%n);

}
                                        //вернуть действительное 0<=fRandom()<1
double RandomNumber::fRandom (void)
{
    return Random(maxshort)/double(maxshort);
}
```

Рисунок 1 Листинг программы

```
Spisok:8  2  5  3  8  4  1  1  10  6
Input key:8
Value  8  appears 2 once in this list
Press any key to continue
```

Рисунок 2 Результаты работы программы

Приложение Б

Пример программы вставки элементов в list<T> с помощью STL

Программа использует шаблон класса list<T> из библиотеки для работы с двунаправленным списком.

```
// При компиляции должна быть включена опция: -GX
//
// insert.cpp : Демонстрирует различные способы вставки элементов в list<T>.
// Функция:  list::insert
#include <list>
#include <iostream>
using namespace std;
typedef list<int> LISTINT;//новый тип на базе list
void main()
{       int rgTest1[] = {5,6,7};       int rgTest2[] = {10,11,12};

LISTINT listInt;
LISTINT listAnother;
LISTINT::iterator i;
                                        // Вставка по одному элементу
listInt.insert (listInt.begin(), 2);
listInt.insert (listInt.begin(), 1);
```

```
listInt.insert (listInt.end(), 3);
                                        // 1 2 3
for (i = listInt.begin(); i != listInt.end(); ++i)
        cout << *i << " ";
        cout << endl;
                                        // Вставка 3 четверок
listInt.insert (listInt.end(), 3, 4);
                                        // 1 2 3 4 4 4
for (i = listInt.begin(); i != listInt.end(); ++i)
        cout << *i << " ";
        cout << endl;
                                        // Вставка массива
listInt.insert (listInt.end(), rgTest1, rgTest1 + 3);
                                        // 1 2 3 4 4 4 5 6 7
for (i = listInt.begin(); i != listInt.end(); ++i)
        cout << *i << " ";
        cout << endl;
                                        // Вставка другого списка LISTINT
listAnother.insert (listAnother.begin(), rgTest2, rgTest2+3);
listInt.insert (listInt.end(), listAnother.begin(), listAnother.end());
                                        // 1 2 3 4 4 4 5 6 7 10 11 12
for (i = listInt.begin(); i != listInt.end(); ++i)
        cout << *i << " ";
        cout << endl;}
```

Рисунок 1 Листинг программы insert.cpp

```
1 2 3
1 2 3 4 4 4
1 2 3 4 4 4 5 6 7
1 2 3 4 4 4 5 6 7 10 11 12
Press any key to continue
```

Рисунок 2 Результаты работы программы

Приложение В

Пример программы сортировки списка с помощью STL

Пример демонстрирует ошибку, которая возникает при обработке списка длиной более 32Кбайт – после сортировки такого списка элементы списка удаляются.

```
//test.cpp
//Compiler options : /GX
#include <list>
#include <iostream>
```

```
void main()
{
std::list<int> mylist;

int i;
int last = 32768;

for (i = 0; i < last; i++)
        mylist.push_back(i);
        std::cout << "Size before sort: " << mylist.size() << "\n";
        mylist.sort();
        std::cout << "Size after sort: " << mylist.size() << "\n";
}
```

Рисунок 1 Листинг программы test.cpp

```
Size before sort: 32768
Size after sort: 0
Press any key to continue
```

Рисунок 2 Результаты работы программы

Приложение Г

Таблица 1 Набор операций, которые поддерживает класс list<T>

Назначение	Функция-член
1	2
Вставить в начале	push_front
Вставить в конце	push_back
Удалить в начале	pop_front
Удалить в конце	pop_back
Вставить в любом месте	insert
Удалить в любом месте	erase
Сортировать	sort
Удалить повторяющиеся одинаковые элементы	unique
Вставка одного или последовательности элементов или целого списка в другой список	splice
Удалить все элементы списка с заданным значением	remove
Заменить последовательность на обратную ей	reverse
Объединение	merge

Приложение Д

Таблица 1 Алгоритмы, применимые к классу list<T>

Назначение	Алгоритм
Сортировка	Алгоритм sort не работает со списками
Поиск	find, find_if, find_first_of, find_end, count, for_each, adjacent_find, mismatch, equal, search,
Объединение	merge
Поменять значения двух последовательностей	swap
Заменить последовательность на обратную ей	reverse
Копировать	copy

Приложение Е

Иерархия контейнерных классов STL

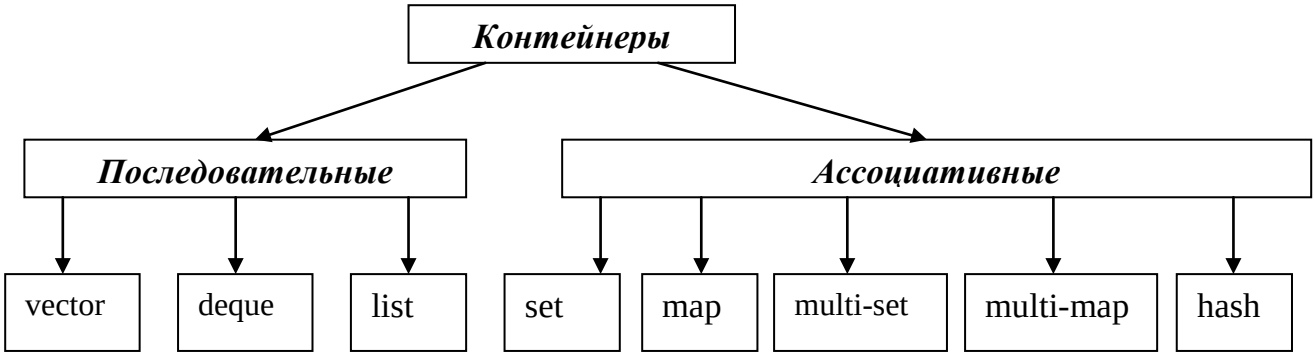

Рисунок 1 Иерархия контейнерных классов STL