## Практическая работа №6. Связные списки

***Задание 6.1***

1. Дан базовый класс Связный Список. Ввести в него деструктор и перегрузить оператор [ ]. Унаследовать от него класс Односвязный Стек и переопределить операции и добавления push() в конец списка и удаления pop() с конца (значение удалённого элемента вернуть как результат функции). Информация в элемент структуры вводится со стандартного потока ввода (предусмотреть автоматическое заполнение). Поля классов Элемент, Связный Список и его наследников должны быть защищёнными (protected).

Унаследовать от класса Связный Список класс Односвязная Очередь (Queue) и переопределить операции добавления push() в конец списка и удаления pop() с начала (значение удалённого элемента вернуть как результат функции).

Унаследовать от классов Односвязный Стек и Односвязная Очередь новый класс StackQueue с ключевым словом protected (см. рис. 6.1). Добавить открытые методы push_back(), push_front(), pop_back(), pop_front() – для добавления и удаления элементов с начала и конца списка. Во всех функциях, где это возможно, вызывать функции одного из родительских классов.

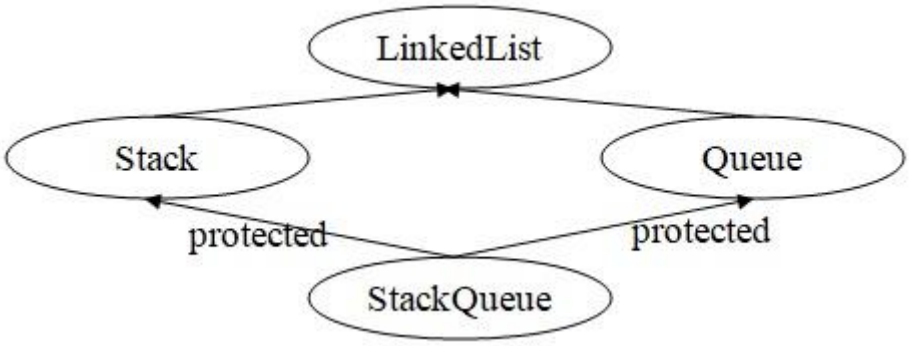


*Рисунок 6.1. Схема наследования односвязных списков для пункта 6.1*

2. Унаследовать класс Двусвязный Стек от Односвязного Стека. Переопределить функции push(), pop(), учитывая эффективность выполнения операций. Перегрузить операцию вывода в поток << и индексации [i]. Написать функцию вставки нового элемента в произвольное место списка insert(), удаления remove(), поиска элемента find() по значению. Реализовать указанный вид поиска: под функцией фильтра предполагается поиск элементов, которые удовлетворяют условию >, <, под поиском – только = =. Фильтр возвращает новый список элементов, удовлетворяющих условию.

Функцию фильтра и поиска реализовать в итеративной и рекурсивной формах. Класс, объекты которого хранятся в списке, указан в таблице 6.1.

3. Унаследовать от класса, реализованного в пункте 2, свой класс с ключевым словом protected. Заменить в нём открытый интерфейс: работа функций push() / pop() и других указана в вариантах в таблице 6.1.

Таблица 6.1. Классы, экземпляры которых необходимо хранить в списке и тип списка

| Вариант | Структура, тип списка |
|---|---|
| 1 | Структура «Государство».<br>Минимальный набор полей: название, столица, язык, численность населения, площадь.<br>Добавление: в начало<br>Удаление: из начала<br>Поиск по названию страны, фильтр по площади. |
| 2 | Структура «Человек».<br>Минимальный набор полей: фамилия, имя, пол, рост, вес, дата рождения, телефон, адрес.<br>Добавление: в конец<br>Удаление: из начала<br>Поиск по фамилии, фильтр по дате рождения. |
| 3 | Структура «Школьник».<br>Минимальный набор полей: фамилия, имя, пол, класс, дата рождения, адрес.<br>Добавление: в начало<br>Удаление: с конца<br>Поиск по фамилии, фильтр по классу. |
| 4 | Структура «Покупатель».<br>Минимальный набор полей: фамилия, имя, город, улица, номера дома и квартиры, номер счёта, средний сумма чека.<br>Добавление: в начало<br>Удаление: из начала<br>Поиск по фамилии, фильтр по средней сумме чека. |
| 5 | Структура «Пациент».<br>Минимальный набор полей: фамилия, имя, дата рождения, телефон, адрес, номер карты, группа крови.<br>Добавление: в начало |

| | |
|---|---|
| | Удаление: с конца<br>Поиск по фамилии, фильтр по группе крови. |
| 6 | Структура «Команда».<br>Минимальный набор полей: название, город, число побед, поражений, ничьих, количество очков.<br>Добавление: в конец<br>Удаление: из начала<br>Поиск по названию, фильтр по числу побед. |
| 7 | Структура «Стадион».<br>Минимальный набор полей: название, виды спорта, год постройки, вместимость, количество арен.<br>Добавление: в начало<br>Удаление: из начала<br>Поиск по названию, фильтр по году постройки. |
| 8 | Структура «Автовладелец».<br>Минимальный набор полей: фамилия, имя, номер автомобиля, дата рождения, номер техпаспорта.<br>Добавление: в конец<br>Удаление: из начала<br>Поиск по номеру техпаспорта, фильтр по фамилии. |
| 9 | Структура «Автомобиль».<br>Минимальный набор полей: марка, цвет, серийный номер, количество дверей, год выпуска, цена.<br>Добавление: в начало<br>Удаление: с конца<br>Поиск по серийному номеру, фильтр по цене. |
| 10 | Структура «Фильм».<br>Минимальный набор полей: фамилия, имя режиссёра, название, страна, год выпуска, стоимость, доход.<br>Добавление: в конец<br>Удаление: из начала<br>Поиск по названию, фильтр по доходу. |
| 11 | Структура «Альбом».<br>Минимальный набор полей: имя или псевдоним исполнителя, название альбома, количество композиций, год выпуска.<br>Добавление: в начало<br>Удаление: из начала |

| | |
|---|---|
| | Поиск по названию альбома, фильтр по имени. |
| 12 | Структура «Матрица».<br>Минимальный набор полей: размерности, коэффициенты матрицы.<br>Добавление: в начало<br>Удаление: из начала<br>Поиск по следу, фильтр по размерности.<br>Дополнительная функция: вычисление определителя и следа, произведения. |
| 13 | Структура «Вектор».<br>Минимальный набор полей: размерность, коэффициенты вектора.<br>Добавление: в начало<br>Удаление: с конца<br>Поиск по длине (норме), фильтр по размерности.<br>Дополнительная функция: вычисление длины вектора, скалярного произведения двух векторов. |
| 14 | Структура «Определённый интеграл».<br>Минимальный набор полей: указатель на подынтегральную функцию, шаг вычисления (точность), пределы интегрирования.<br>Добавление: в конец<br>Удаление: с начала<br>Фильтр по значению интеграла.<br>Дополнительная функция: приближённое вычисление значения интеграла. |
| 15 | Структура «Уравнение».<br>Минимальный набор полей: указатель на функцию f(x) в уравнении, шаг вычисления (точность), левая и правая границы области, в которой производится поиск решения уравнения.<br>Добавление: в начало<br>Удаление: из начала<br>Поиск по решениям уравнения.<br>Дополнительная функция: приближённое вычисление решения уравнения. |
| 16 | Структура «Государство».<br>Минимальный набор полей: название, столица, язык, численность населения, площадь.<br>Добавление: в начало<br>Удаление: с конца |

| | |
|---|---|
| | Поиск по численности населения, фильтр по языку. |
| 17 | Структура «Человек».<br>Минимальный набор полей: фамилия, имя, пол, рост, вес, дата рождения, телефон, адрес.<br>Добавление: в начало<br>Удаление: с конца<br>Поиск по фамилии, фильтр по адресу. |
| 18 | Структура «Школьник».<br>Минимальный набор полей: фамилия, имя, пол, класс, дата рождения, адрес.<br>Добавление: в начало<br>Удаление: из начала<br>Поиск по фамилии, фильтр по дате рождения. |
| 19 | Структура «Покупатель».<br>Минимальный набор полей: фамилия, имя, город, улица, номера дома и квартиры, номер счёта.<br>Добавление: в конец<br>Удаление: из начала<br>Поиск по номеру счета, фильтр по городу. |
| 20 | Структура «Пациент».<br>Минимальный набор полей: фамилия, имя, дата рождения, телефон, адрес, номер карты, группа крови.<br>Добавление: в начало<br>Удаление: с конца<br>Поиск номеру карты, фильтр по группе крови.. |
| 21 | Структура «Команда».<br>Минимальный набор полей: название, город, число побед, поражений, ничьих, количество очков.<br>Добавление: в начало<br>Удаление: из начала<br>Поиск по названию, фильтр по числу побед. |
| 22 | Структура «Автовладелец».<br>Минимальный набор полей: фамилия, имя, номер автомобиля, дата рождения, номер техпаспорта.<br>Добавление: в конец<br>Удаление: из начала |

| | |
|---|---|
| | Поиск по номеру автомобиля, фильтр по имени. |
| 23 | Структура «Государство».<br>Минимальный набор полей: название, столица, язык, численность населения, площадь.<br>Добавление: в начало<br>Удаление: с конца<br>Поиск по столице, фильтр по численности населения. |
| 24 | Структура «Программист».<br>Минимальный набор полей: фамилия, имя, email, skype, telegram, основной язык программирования, текущее место работы, уровень (число от 1 до 10).<br>Добавление: в начало<br>Удаление: из начала<br>Поиск по email, фильтр по уровню. |
| 25 | Структура «Спортсмен».<br>Минимальный набор полей: фамилия, имя, возраст, гражданство, вид спорта, количество медалей.<br>Добавление: в конец<br>Удаление: из начала<br>Поиск по фамилии, фильтр по числу медалей. |
| 26 | Структура «Полином».<br>Минимальный набор полей: степень полинома, коэффициенты<br>Добавление: в начало<br>Удаление: с конца<br>Поиск по коэффициентам, фильтр по степени полинома.<br>Дополнительная функция: поиск корня полинома на указанном отрезке. |
| 27 | Структура «Профиль в соц.сети».<br>Минимальный набор полей: псевдоним, адрес страницы, возраст, количество друзей, интересы, любимая цитата.<br>Добавление: в начало<br>Удаление: из начала<br>Поиск по псевдониму, фильтр по количеству друзей. |
| 28 | Структура «Велосипед».<br>Минимальный набор полей: марка, тип, тип тормозов, количество колес, диаметр колеса, наличие амортизаторов, детский или взрослый. |

| | |
|---|---|
| | Добавление: в начало<br>Удаление: с конца<br>Поиск по марке, фильтр по диаметру колеса. |
| 29 | Структура «Ноутбук».<br>Минимальный набор полей: производитель, модель, размер экрана, процессор, количество ядер, объем оперативной памяти, объем диска, тип диска, цена.<br>Добавление: в конец<br>Удаление: из начала<br>Поиск по марке, фильтр по цене. |
| 30 | Структура «Смартфон».<br>Минимальный набор полей: марка, размер экрана, количество камер, объем аккумулятора, максимальное количество часов без подзарядки, цена.<br>Добавление: в начало<br>Удаление: из начала<br>Поиск по марке, фильтр по размеру экрана. |

***Задание 6.2***

1. Кроме фильтра по указанным полям, необходимо реализовать универсальный фильтр по произвольному полю, который принимает указатель на функцию. Эта функция возвращает 1, если элемент удовлетворяет условию и 0 – в противном случае.

2. Выделите память для списка динамически (используя указатель на базовый класс), а затем, в конце работы кода, освободите её. Проверьте правильную работу деструкторов базового и производного классов (используйте виртуальный деструктор). Преобразуйте указатель на базовый класс в указатель типа производного класса с помощью оператора dynamic_cast. Проверьте работу деструкторов в случае выполнения операции освобождения памяти для преобразованного указателя.

3. Напишите функции сохранения списка файл* save() и извлечения списка из файла* load(). Для ввода/вывода элемента списка в/из файла используйте переопределённые функции ввода/вывода в поток (сделайте различие между вводом/выводом в консоль и вводом/выводом в файл).

4. Введите манипулятор, форматирующий вывод, и примените его при выводе содержимого контейнера из задания 6.1. Заполните контейнер произвольными числами (см. таблицу 6.2).

Таблица 6.2. Манипуляторы, форматирующие вывод

| Вариант | Манипуляторы вывода |
|---|---|
| 1. | Вывод элементов в 1) научном виде, 2) 8-ричной системе счисления |
| 2. | Вывод в научном виде, ширина поля – 5 символов, 2 знака после запятой |
| 3. | Вывод в 8-ричной системе счисления, выравнивание по левому краю |
| 4. | Установить ширину поля при выводе чисел в 5 символов. |
| 5. | Вывод в 16-ричной системе счисления |
| 6. | Вывод с расстановкой табуляций в 16-ричном виде |
| 7. | Вывод в 16-ричной системе счисления, выравнивание по правому краю |
| 8. | 3 знака после запятой, не более 7 символов на число |
| 9. | Символ заполнения - ‘%’, выравнивание по правому краю |
| 10. | Вывод в 16-ричном виде и с символом заполнения ‘&’ |
| 11. | Установить ширину поля при выводе чисел в 5 символов. |
| 12. | Вывод в виде 16-ричных чисел в научной форме |
| 13. | Установить ширину поля при выводе чисел в 8 символов, точность – 3 символа после запятой. |
| 14. | В 8-ричной системе счисления, не более 7 символов на действительное число |
| 15. | Выравнивание вывода по левому краю в верхнем регистре |
| 16. | Показывать десятичную точку, 2 знака (как минимум) до неё и 2 знака после |
| 17. | Вывод в научном виде, точность – 5 знаков после запятой |
| 18. | Каждое число содержит 2 знака (как минимум) до запятой и 2 знака после |
| 19. | Вывод в 16-ричной системе счисления, с символом заполнения ‘_’ и шириной поля в 10 символов при выводе числа |
| 20. | Ширина поля при выводе – 5 символов, число символов после запятой – 2. |
| 21. | Вывод в 16-ричной системе счисления, не более 3 знаков после запятой |

Код 6.1. Пример базового класса для связного списка LinkedList и наследования класса Стек

```
template<class T>
class Element
{
 //protected:
public:
 //переместить в protected
 Element* next;
 Element* prev;
 T info;

 Element(T data)
 {
      next = prev = NULL;
      info = data;
 }

 Element(Element* Next, Element* Prev, T data)
 {
      next = Next;
      prev = Prev;
      info = data;
 }

 Element(const Element& el)
 {
      next = el.next;
      prev = el.prev;
      info = el.info;
 }

 template<class T1>
 friend ostream& operator<<(ostream& s, Element<T1>& el);

};

template<class T1>
ostream& operator<<(ostream& s, Element<T1>& el)
{
 s << el.info;
 return s;
}

template<class T>
class LinkedList
{
 //protected:
public:
 //переместить в protected
 Element<T>* head;
 Element<T>* tail;
 int count;

 LinkedList()
```

```
        {
            head = tail = NULL;
            count = 0;
        }

        virtual Element<T>* pop() = 0;
        virtual Element<T>* push(T value) = 0;

        virtual Element<T>& operator[](int index) = 0;

        virtual bool isEmpty() { return (LinkedList<T>::count == 0);
}

        template<class T1>
        friend ostream& operator<<(ostream& s, LinkedList<T1>& el);

        virtual ~LinkedList()
        {
            cout << "\nBase class destructor";
            //дописать деструктор базового класса
            head = NULL; tail = NULL;
        }
       };

       template<class T1>
       ostream& operator<<(ostream& s, LinkedList<T1>& el)
       {
        …
       }

template<class T, int N = 20>
class Stack : public LinkedList<T>
{
public:
        Stack<T, N>() : LinkedList<T>()
        {
            if (N > 0)
                for (int i = 0; i < N; i++)
                    push(0);
        }

        virtual Element<T>* push(T value)
        {
            if (LinkedList<T>::head == NULL) //if(count==0)
            {
                //пустой список
                LinkedList<T>::tail = new Element<T>(value);
                LinkedList<T>::head = LinkedList<T>::tail;
            }
            else
            {
                //элементы уже есть
                LinkedList<T>::tail->next = new Element<T>(value);
```

```
                //LinkedList<T>::tail->next->prev =
LinkedList<T>::tail;
                LinkedList<T>::tail = LinkedList<T>::tail->next;
           }
           LinkedList<T>::count++;
           return LinkedList<T>::tail;
     }

     virtual Element<T>* pop()
     {
           //пустой список
           if (LinkedList<T>::tail == NULL)
                return NULL;
           Element<T>* res = LinkedList<T>::tail;
           //один элемент
           if (LinkedList<T>::head == LinkedList<T>::tail)
                LinkedList<T>::head = LinkedList<T>::tail = NULL;
           else
           {
                Element<T>* current;
                for (current = LinkedList<T>::head;
                     current->next != LinkedList<T>::tail; current
= current->next);
                LinkedList<T>::tail = current;
                LinkedList<T>::tail->next = NULL;
           }
           LinkedList<T>::count--;
           return res;
     }


     virtual ~Stack() { cout << "\nStack class destructor"; }
    };

    ostream& operator<<(ostream& s, my_class& value)
    {
     s << value.data;
     return s;
    }

    int main()
    {
     if (true)
     {
           Stack<double, 20> S;
           for (int i = 0; i < 10; i++)
                S.push(i);
           S.insert(3.5, S.head->next->next->next);
           cout << S;
           cout << "\n";
           //cout<<S.Find_R(5.5, S.head);
     }
    return 0;}
```