

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Рязанский государственный радиотехнический университет
имени В. Ф. Уткина»

Кафедра «Вычислительная и прикладная математика»

Отчет
по лабораторной работе № 5
по дисциплине
«Объектно-ориентированное программирование»
на тему
«Шаблонные классы»

Выполнил:
студент гр. 143
Вербицкая И. С.

Проверил:
Антипов О.В.

Рязань 2022

Задание

В данной лабораторной работе необходимо разработать шаблонный класс в соответствии с вариантом задания. Любой класс должен иметь конструктор копии и перегруженный оператор =. Также класс должен содержать перегрузку оператора << для класса `iostream`, чтобы его можно было выводить в консоль через стандартный поток вывода `cout` с указанием количества элементов и их значений. В главной функции `main` обеспечить консольный интерфейс для тестирования всех функций шаблонного класса с типами `int`, `float`, `char*`, `struct Vec2 {float x; float y;}` (аналогично лаб 1) Non-type параметр при тестировании класса можно задавать через константу.

Важно: методы, имеющие в качестве параметров объекты того-же типа, что и сам описываемый шаблонный класс, используют с ним одинаковое значение non-type параметра.

5. **Односвязный список**, динамическая структура данных, где через non-type параметр задаётся максимально допустимая характеристика добавляемого элемента (для типов `int` и `float` это модуль значения, для `char*` - длина строки, для `Vec2` – длина вектора). Т.е. элементы с характеристикой, превышающей заданное значение нельзя добавить в список.

Необходимые методы:

getLength() – получить текущее кол-во элементов

isExist(elem) – получить вхождение элемента в список (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки)

insert(elem,n) – добавить элемент на позицию `n`

remove(elem,n) – удалить элемент на позиции `n`

Перегрузить следующие операции:

+ – добавить элемент в начало списка.

-- – удалить все вхождения элемента из списка (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки)

-- – удалить первый элемент из начала списка.

[] – доступ к элементу на заданной позиции.

Описание структуры программы

Созданный для работы с динамическим односвязным списком класс `List` – шаблонный класс-потомок абстрактного класса `BaseList`. Для `List` имеется несколько реализаций для различных типов, а также общая реализация. Шаблон абстрактного класса `BaseList` не содержит специализаций. Кроме того, дополнительно есть шаблон структуры `Node`, представляющей из себя структуру узла односвязного списка.


Рассмотрим состав структур и классов подробнее.

Структура Node

Шаблон этой структуры принимает некоторый тип данных `T`.

Поля структуры:

 `T Val` – значение узла


 `Node<T>* Head` - указатель на следующий узел

Класс BaseList

Предусмотрена лишь одна общая реализация шаблона, принимающая тип данных `T` и некоторое целочисленное значение `MaxSize`.

Поля класса:

 `int Len` – количество элементов списка

 `Node<T>* Head` - указатель на первый элемент списка

Методы класса:

- ✚ `bool Empty() const` – возвращает `True`, если список пуст
- ✚ `virtual bool isCompatible (Node<T>* Elem)` – возвращает `True`, если элемент соответствует значению `MaxSize`, это чистая виртуальная функция, которая реализуется в классе `List`
- ✚ `virtual bool isEqual (T a, T b)` – возвращает `True`, если элементы равны
- ✚ `virtual void DelAddr(Node<T>* Addr)` – удаляет из списка элемент с адресом `Addr`
- ✚ `virtual void CreateValue (T& Value, T Elem)` – записывает в `Value` значение `Elem`
- ✚ `BaseList()` – конструктор класса
- ✚ `virtual ~BaseList()` – деструктор класса
- ✚ `int getLength() const` – возвращает `Len`
- ✚ `bool isIndex(int ind) const` – возвращает `True`, если индекс `ind` входит в пределы списка (больше нуля и меньше его длины)
- ✚ `void Clear()` – полностью очищает список вместе с памятью, выделенной под элементы
- ✚ `bool isExist(T Elem)` – возвращает `True`, если элемент присутствует в списке
- ✚ `virtual void print ()` – выводит список в консоль
- ✚ `T& operator- (T Elem)` – удаляет всех вхождения элемента в список
- ✚ `T& operator-- ()` – удаляет первый элемент списка
- ✚ `T& operator+ (T Elem)` – добавляет элемент в начало списка
- ✚ `void insert(T Elem, int n)` – добавляет элемент в список по его индексу
- ✚ `void remove(T Elem, int n)` – удаляет элемент из списка по его индексу
- ✚ `T& operator[](int index)` – возвращает значение элемента по его индексу

Класс List

Шаблон этого класса, так же, как его родитель `BaseList`, принимает тип данных `T` и некоторое целочисленное значение `MaxSize`.

Для `List` предусмотрены следующие реализации:

- ✚ Общая реализация (базовый шаблон)
- ✚ Реализация для типа `Vec2` (частичная специализация шаблона) (дополнительно к специализации – структура `Vec2` – структура вектора)
- ✚ Реализация для типа `char*` (частичная специализация шаблона)

Класс List (общая реализация)

Методы класса:

- ✚ `bool isCompatible (Node<T>* Elem) override` – перегрузка метода `BaseList`, вычисляющая модуль значения элемента и сравнивающая его со значением `MaxSize`

Структура Vec2

Поля структуры:

- ✚ `int x, int y` – координаты вектора

Методы структуры:

- ✚ `Vec2& operator= (Vec2 Value)` – перегрузка оператора копирования
- ✚ `bool operator== (Vec2 Value), bool operator!= (Vec2 Value)` – перегрузка операторов `==` и `!=`

Кроме того, для `Vec2` реализована перегрузка оператора `<<` для вывода вектора в стандартном потоке вывода, однако она вынесена за пределы структуры:

- ✚ `ostream& operator<<(ostream& os, const Vec2& vec)`

Класс List (реализация для типа Vec2)

Методы класса:

- ✚ `bool isCompatible (Node<T>* Elem) override` – перегрузка метода `BaseList`, вычисляющая длину вектора и сравнивающая ее со значением `MaxSize`

Класс List (реализация для типа `char*`)

Методы класса:

- ✚ `~List()` – измененный деструктор класса, предусматривающий также очистку динамической памяти, выделенной под строки, хранящиеся в элементах, также выделенных динамически.
- ✚ `void ClearContent()` – очищает память, выделенную под строки во всех элементах списка
- ✚ `bool isCompatible (Node<char*>* Elem) override` - перегрузка метода `BaseList`, вычисляющая длину строки и сравнивающая ее со значением `MaxSize`
- ✚ `bool isEqual (char* a, char* b) override` – сравнивает содержимое двух строк, и если оно совпадает возвращает `True`
- ✚ `void CreateValue (char*& Value, char* Elem) override` – записывает в `Value` содержимое строки `Elem`
- ✚ `void DelAddr(Node<char*>* Addr) override` – перегрузка метода удаления элемента по его адресу, предусматривающая также очистку памяти, которая выделяется на хранение самих строк в содержимом элементов списка

Листинг программы:

Файл BaseList.h

```
1  #pragma once
2  #include <iostream>
3  #include <math.h>
4
5  using namespace std;
6
7  //структура узла односвязного списка
8  template <typename T>
9  struct Node
10 {
11     T Val; //содержимое элемента
12     Node* Next; //указатель на следующий узел
13 };
14
15 //односвязный динамический список (общий шаблон)
16 template <class T, int MaxSize>
17 class BaseList
18 {
19 protected:
20     int Len; //кол-во элементов списка
21     Node<T>* Head; //первый элемент списка
22
23     bool Empty() const { return Head==nullptr; } //пуст ли список?
24     virtual bool isCompatible (Node<T>* Elem) = 0; //подходит ли элемент под ограничение MaxSize?
25     virtual bool isEqual (T a, T b) { return a==b; } //равны ли элементы?
26
27     //удаляет элемент по его адресу
28     virtual void DelAddr(Node<T>* Addr) {
29         if (Addr!=nullptr) {
30             if (Addr==Head) {
31                 Head=Addr->Next;
32                 delete Addr;
33                 Addr = nullptr;
34                 Len--;
35             }
36             else {
37                 Node<T>* Dop;
38                 Dop=Head;
39                 while (Dop->Next!=Addr)
40                     Dop=Dop->Next;
41                 Dop->Next=Addr->Next;
42                 delete Addr;
43                 Addr = nullptr;
44                 Len--;
45             }
46         }
47     }
48
49     //конструктор копирования для большинства типов
50     virtual void CreateValue (T& Value, T Elem) { Value = Elem; }
51 }
```

```

52 public:
53
54     BaseList() { Head = nullptr; Len = 0; }
55     virtual ~BaseList() { Clear(); }
56
57     //кол-во элементов в списке
58     int getLength() const { return Len; }
59
60     //не выходит ли индекс за пределы списка?
61     bool isIndex(int ind) const { return (ind<Len) && (ind>=0); }
62
63     //полная очистка списка
64     void Clear() {
65         while (!Empty())
66             DelAddr(Head);
67         Len = 0;
68     }
69
70     //проверка вхождения элемента в список
71     bool isExist(T Elem) {
72         bool F = false;
73         Node<T>* Dop;
74         Dop=Head;
75         while (Dop!=nullptr) {
76             //if (Dop->Val==Elem)
77             if (isEqual(Dop->Val,Elem))
78                 F = true;
79             Dop = Dop->Next;
80         }
81         return F;
82     }
83
84     //вывод содержимого списка
85     virtual void print () {
86         cout << "\n| ОДНОСВЯЗНЫЙ СПИСОК:" << endl;
87         Node<T>* Dop;
88         Dop=Head;
89         for(int i = 0; i < Len; ++i) {
90             cout << "| " << i+1 << " | " << Dop->Val << "\n";
91             Dop=Dop->Next;
92         }
93         cout << "| КОЛ-ВО ЭЛЕМЕНТОВ: " << Len << endl;
94         cout << "| МАКС ХАРАК-ТИКА: " << MaxSize << endl;
95     }
96
97     //удалить все вхождения элемента из списка
98     T& operator- (T Elem) {
99         while (isExist(Elem))
100             for (int i=0; i<Len; i++)
101                 remove(Elem,i);
102     }
103

```



```

104 //удалить первый элемент из начала списка
105 T& operator-- () { if (!Empty()) remove(Head->Val,0); }
106
107 //добавить элемент в начало списка
108 T& operator+ (T Elem) { insert(Elem,0); }
109
110 //добавить элемент на позицию n
111 void insert(T Elem, int n) {
112     Node<T>* Dop;
113     Dop=Head;
114     Node<T>* New = new Node<T>; //выделяем память под узел
115     CreateValue(New->Val,Elem);
116     //New->Val = Elem;
117     Len++;
118     if (isCompatible(New)) {
119         if (Empty()) { //вставка в пустой список
120             New->Next=nullptr;
121             Head = New;
122         }
123         else if (n==Len-1) { //вставка в конец списка
124             while ((Dop->Next)!=nullptr) Dop=Dop->Next;
125             New->Next=nullptr;
126             Dop->Next=New;
127         }
128         else if (n==0) { //вставка в начало списка
129             New->Next = Head;
130             Head = New;
131         }
132         else if ((n<Len-1)&&(n>0)) { //вставка в середину списка
133             int i=0;
134             while (i<n-1) { //приходим к предшествующему n элементу
135                 Dop=Dop->Next;
136                 i++;
137             }
138             New->Next = Dop->Next; //Наш новый эл указывает на эл, идущие после n-го
139             Dop->Next = New; //Эл перед n-ым начинает указывать на новый эл
140         }
141         else { //попытка вставки на недопустимый индекс
142             delete New;
143             New=nullptr;
144             Len--;
145         }
146     }
147     else { //попытка вставки недопустимого по MaxSize значения
148         delete New;
149         New=nullptr;
150         Len--;
151     }
152 }
153

```

```

154 //удалить элемент с позиции n
155 void remove(T Elem, int n) {
156     if (isExist(Elem)) {
157         int D = 0;
158         Node<T>* Dop;
159         Dop=Head;
160
161         while (Dop!=nullptr) {
162             //if ((D==n)&&(Dop->Val==Elem)) {
163             if ((D==n)&&(isEqual(Dop->Val,Elem))) {
164                 DelAddr(Dop);
165                 break;
166             }
167             D++;
168             Dop = Dop->Next;
169         }
170     }
171 }
172
173 //доступ к элементу на заданной позиции
174 T& operator[](int index) {
175     if ((index<Len) && (index>=0)) {
176         Node<T>* Dop;
177         Dop=Head;
178         for (int i=0;i<index;i++)
179             Dop=Dop->Next;
180         return Dop->Val;
181     }
182 }
183
184 };
185

```


Файл List.h

```
1  #include "BaseList.h"
2  #include <strings.h>
3
4  using namespace std;
5
6  //----- int, float -----
7
8  template <class T, int MaxSize>
9  class List: public BaseList<T,MaxSize>
10 {
11     //подходит ли элемент под ограничение MaxSize?
12     bool isCompatible (Node<T>* Elem) override {
13         return fabs(Elem->Val)<=static_cast<float>(MaxSize);
14     }
15 };
16
17 //----- Vec2 -----
18 struct Vec2 {
19     int x;
20     int y;
21
22     //конструктор копирования
23     Vec2& operator= (Vec2 Value) {
24         x = Value.x;
25         y = Value.y;
26         return *this;
27     }
28
29     bool operator== (Vec2 Value) {
30         return (x == Value.x) && (y == Value.y);
31     }
32
33     bool operator!= (Vec2 Value) {
34         return (x != Value.x) || (y != Value.y);
35     }
36 };
37
38 // перегрузка оператора << класса ostream для Vec2
39 // для вывода в стандартном потоке вывода
40 ostream& operator<<(ostream& os, const Vec2& vec) {
41     return os << "(" << vec.x << "; " << vec.y << ")";
42 }
43
44
45 template <int MaxSize>
46 class List<Vec2,MaxSize>: public BaseList<Vec2,MaxSize>
47 {
48     //подходит ли элемент под ограничение MaxSize?
49     bool isCompatible (Node<Vec2>* Elem) override {
50         int X = Elem->Val.x;
51         int Y = Elem->Val.y;
52         int R = MaxSize;
53         return X*X+Y*Y<=R*R;
```

```

54 }
55 };
56
57 //----- char * -----
58 template <int MaxSize>
59 class List<char*,MaxSize>: public BaseList<char*,MaxSize>
60 {
61 public:
62     ~List() { ClearContent(); }
63
64 private:
65     //для всех элементов списка удаляет память,
66     //выделенную под строки, хранимые в эл-тах
67     void ClearContent() {
68         if (!this->Empty()) {
69             Node<char*>* Dop;
70             Dop = this->Head;
71             while (Dop!=nullptr) {
72                 delete Dop->Val;
73                 Dop->Val = nullptr;
74                 Dop = Dop->Next;
75             }
76         }
77     }
78
79     //равны ли элементы?
80     bool isCompatible (Node<char*>* Elem) override { return strlen(Elem->Val) < MaxSize; }
81     bool isEqual (char* a, char* b) override { return strcmp(a,b)==0; }
82     //bool operator== (char* a, char* b) {return strcmp(a,b)==0}
83
84     //конструктор копии
85     void CreateValue (char*& Value, char* Elem) override {
86         Value = new char[strlen(Elem)];
87         if(Elem) strcpy(Value, Elem);
88         else strcpy(Value, " ");
89     }
90     /*char* operator= (const char* value) {
91         printf("OPA COPY");
92         char* S = new char[strlen(value)];
93         if(value) strcpy(S, value);
94         else strcpy(S, " ");
95         return S;
96     }*/
97
98     void DelAddr(Node<char*>* Addr) override {
99         //cout << "Удаляем " << Addr->Val << endl;
100         delete Addr->Val;
101         Addr->Val = nullptr;
102         BaseList<char*,MaxSize>::DelAddr(Addr);
103     }
104 };
105

```

```

1  #pragma once
2  #include <iostream>
3  #include "List.h"
4
5  #define MAXSTR 256
6  const int MAXSIZE = 16;
7
8  using namespace std;
9
10 //Предобъявления
11 void MainMenu();
12 void CreateMenu(int Key);
13
14 template <class T, int MaxSize>
15 void ListMenu(List<T,MaxSize>& MyList);
16
17 template <class T, int MaxSize>
18 void InputElementMenu(List<T,MaxSize>& MyList, int Key);
19
20 //Стартовое меню
21 void MainMenu() {
22     int Key; //номер вводимой команды
23     while (true) {
24         system("cls");
25         cout << "\n----- СОЗДАНИЕ ШАБЛОННОГО СПИСКА -----\n" << endl;
26         cout << " Список какого типа будем тестировать?" << endl;
27         cout << " | 1 |                int                |" << endl;
28         cout << " | 2 |                float               |" << endl;
29         cout << " | 3 |                Vec2                 |" << endl;
30         cout << " | 4 |                char*                |" << endl;
31         cout << " | 5 |    Никакого! (Завершить работу)    |" << endl;
32         cout << "\nВведите номер команды: "; cin >> Key;
33
34         if ((Key>=1)&&(Key<=4))
35             CreateMenu(Key);
36         else if (Key==5)
37             exit(0);
38         else {
39             puts("Такой команды не предусмотрено!");
40             system("pause");
41         }
42     }
43 }
44

```



```

45 //Создание списка
46 void CreateMenu(int Key) {
47     switch (Key) {
48         case 1: { List<int, MAXSIZE> L; ListMenu(L); break; }
49         case 2: { List<float,MAXSIZE> L; ListMenu(L); break; }
50         case 3: { List<Vec2, MAXSIZE> L; ListMenu(L); break; }
51         case 4: { List<char*,MAXSIZE> L; ListMenu(L); break; }
52     }
53 }
54 }
55
56 //Работа с шаблонным списком
57 template <class T, int MaxSize>
58 void ListMenu(List<T,MaxSize>& MyList) {
59     int n;
60     int Key;
61     while (true) {
62         system("cls");
63         cout << "\n----- РАБОТА С ШАБЛОННЫМ СПИСКОМ ----- \n" << endl;
64         cout << "Какую команду вы хотите выполнить со списком?" << endl;
65         cout << " 1 | getLength() | узнать текущее количество элементов | " << endl;
66         cout << " 2 | isExist(elem) | узнать, есть ли элемент в списке | " << endl;
67         cout << " 3 | insert(elem,n) | добавить элемент elem на позицию n | " << endl;
68         cout << " 4 | remove(elem,n) | удалить элемент elem с позиции n | " << endl;
69         cout << " 5 | + | добавить какой-либо элемент в начало списка | " << endl;
70         cout << " 6 | - | удалить все вхождения какого-то элемента | " << endl;
71         cout << " 7 | -- | удалить первый элемент из начала списка | " << endl;
72         cout << " 8 | [] | узнать значени элемента по его индексу | " << endl;
73         cout << " 9 | Вернуться к меню создания списка | " << endl;
74         MyList.print();
75         cout << "\nВведите номер команды: "; cin >> Key;
76         switch (Key) {
77             case 1: { printf("\nКол-во элементов в списке: %d\n", MyList.getLength()); break; }
78             //case (2,3,4,5,6): case (2-6):
79             case 2: case 3: case 4: case 5: case 6:
80                 { InputElementMenu(MyList,Key); break; }
81             case 7: { --MyList; MyList.print(); break; }
82             case 8: {
83                 cout << "n = "; cin >> n;
84                 if (MyList.isIndex(n-1)) cout << "\nЭлемент № " << n << ": " << MyList[n-1] << endl;
85                 else cout << "\nИндекс выходит за пределы списка!" << endl;
86                 break;
87             }
88             case 9: { MyList.Clear(); MainMenu(); break; }
89             default:{ puts("Такой команды не предусмотрено!"); }
90         }
91         printf("\n"); system("pause");

```

```

92     }
93 }
94
95 //Работа с шаблонным списком + ввод элемента (специализации в зависимости от типа элемента)
96 // общая реализация
97 template <class T, int MaxSize>
98 void InputElementMenu(List<T,MaxSize>& MyList, int Key) {
99     //ввод элемента и его индекса (при необходимости)
100     T elem;
101     cout << "\nelem = "; cin >> elem;
102     int n = 0;
103     if ((Key == 3) || (Key == 4)) { cout << "n = "; cin >> n; }
104
105     switch (Key) {
106     case 2: {
107         if (MyList.isExist(elem)) puts("Да, такой элемент есть в списке");
108         else puts("Нет, такого элемента в списке нет");
109         break; }
110     case 3: { MyList.insert(elem,n-1); MyList.print(); break; }
111     case 4: { MyList.remove(elem,n-1); MyList.print(); break; }
112     case 5: { MyList+elem; MyList.print(); break; }
113     case 6: { MyList-=elem; MyList.print(); break; }
114     }
115 }
116
117 //Реализация под ввод Vec2
118 template <int MaxSize>
119 void InputElementMenu(List<Vec2,MaxSize>& MyList, int Key) {
120     //ввод элемента и его индекса (при необходимости)
121     int x; int y; Vec2 elem;
122     cout << "\nelem:\n";
123     cout << "x = "; cin >> x;
124     cout << "y = "; cin >> y;
125     elem = {x,y};
126     int n = 0;
127     if ((Key == 3) || (Key == 4)) { cout << "n = "; cin >> n; }
128
129     switch (Key) {
130     case 2: {
131         if (MyList.isExist(elem)) puts("Да, такой элемент есть в списке");
132         else puts("Нет, такого элемента в списке нет");
133         break; }
134     case 3: { MyList.insert(elem,n-1); MyList.print(); break; }
135     case 4: { MyList.remove(elem,n-1); MyList.print(); break; }
136     case 5: { MyList+elem; MyList.print(); break; }
137     case 6: { MyList-=elem; MyList.print(); break; }
138     }
139 }

```



```

140
141 //Реализация под ввод char*
142 template <int MaxSize>
143 void InputElementMenu(List<char*,MaxSize>& MyList, int Key) {
144     //ввод элемента и его индекса (при необходимости)
145     char* s = new char[MAXSTR];
146     cout << "\nelem = "; cin >> s;
147     int n = 0;
148     if ((Key == 3) || (Key == 4)) { cout << "n = "; cin >> n; }
149
150     switch (Key) {
151     case 2: {
152         if (MyList.isExist(s)) puts("Да, такой элемент есть в списке");
153         else puts("Нет, такого элемента в списке нет");
154         break; }
155     case 3: { MyList.insert(s,n-1); MyList.print(); break; }
156     case 4: { MyList.remove(s,n-1); MyList.print(); break; }
157     case 5: { MyList+s; MyList.print(); break; }
158     case 6: { MyList-s; MyList.print(); break;}
159     }
160     delete s; s = nullptr;
161 }
162
163

```

```

1  #include "Menu.h"
2
3  int main() {
4      system("chcp 1251");
5      MainMenu();
6
7      //работа со строками напрямую
8      /* List<char*,30> L;
9
10     char* s1 = new char[128]; strcpy(s1, "Cmp1");
11     char* s2 = new char[128]; strcpy(s2, "Cmp2");
12     char* s3 = new char[128]; strcpy(s3, "Cmp3");
13
14     L.insert(s1,0); L.insert(s2,0); L.insert(s3,2);
15     L.insert("mda",3); L.insert("mda",4);
16     L.print();
17
18     L.remove(s2,0); L+"teststr"; L-"mda"; --L;
19
20     delete[] s1; delete[] s2; delete[] s3; // Удаляем строки!
21     L.print();
22 */
23
24     //работа с int
25     /* List<int,6> MyList;
26
27     MyList.insert(2,0); MyList.insert(3,1); MyList.insert(4,2);
28     MyList.insert(66,2); MyList.insert(2,5);
29     MyList.print();
30
31     MyList.insert(5,3); MyList.remove(5,3); MyList+6;
32     --MyList; MyList-2;
33     MyList.print();
34 */
35
36     //работа с Vec2
37     /* List<Vec2,4> L;
38
39     L.insert({2,3},0); L.insert({2,1},1); L.insert({2,2},2);
40     L.print();
41
42     Vec2 V = {1,1};
43     L+V; L+V; L-V; |--L;
44     L.print();
45 */
46     return 0;
47 }

```

Результаты работы программы:

В результате выполнения лабораторной было разработано консольное приложение с двумя страницами меню (рисунок 1-2), позволяющими тестировать разработанный шаблон односвязного динамического списка, выбирая один из четырех предложенных типов данных и выполняя над списком различные действия.

```
----- СОЗДАНИЕ ШАБЛОННОГО СПИСКА -----
Список какого типа будем тестировать?
1 | int
2 | float
3 | Vec2
4 | char*
5 | Никакого! (Завершить работу)

Введите номер команды: _

----- РАБОТА С ШАБЛОННЫМ СПИСКОМ -----
Какую команду вы хотите выполнить со списком?
1 | getLength() | узнать текущее количество элементов
2 | isExist(elem) | узнать, есть ли элемент в списке
3 | insert(elem,n) | добавить элемент elem на позицию n
4 | remove(elem,n) | удалить элемент elem с позиции n
5 | + | добавить какой-либо элемент в начало списка
6 | - | удалить все вхождения какого-то элемента
7 | -- | удалить первый элемент из начала списка
8 | [] | узнать значени элемента по его индексу
9 | Вернуться к меню создания списка

Односвязный список:
КОЛ-ВО ЭЛЕМЕНТОВ: 0
МАКС ХАРАК-ТИКА: 16

Введите номер команды: _
```

Рисунок 1 – Главное меню

Рисунок 2 – Меню работы со списком

В качестве примера приведем работу списка, хранящего значения типа `char*` (рисунки3-13)

```
----- СОЗДАНИЕ ШАБЛОННОГО СПИСКА -----
Список какого типа будем тестировать?
1 | int
2 | float
3 | Vec2
4 | char*
5 | Никакого! (Завершить работу)

Введите номер команды: 4_
```

Рисунок 3 – Создание строкового списка

```
----- РАБОТА С ШАБЛОННЫМ СПИСКОМ -----
Какую команду вы хотите выполнить со списком?
1 | getLength() | узнать текущее количество элементов
2 | isExist(elem) | узнать, есть ли элемент в списке
3 | insert(elem,n) | добавить элемент elem на позицию n
4 | remove(elem,n) | удалить элемент elem с позиции n
5 | + | добавить какой-либо элемент в начало списка
6 | - | удалить все вхождения какого-то элемента
7 | -- | удалить первый элемент из начала списка
8 | [] | узнать значени элемента по его индексу
9 | Вернуться к меню создания списка

Односвязный список:
КОЛ-ВО ЭЛЕМЕНТОВ: 0
МАКС ХАРАК-ТИКА: 16

Введите номер команды: 5

elem = строка

Односвязный список:
1 | строка
КОЛ-ВО ЭЛЕМЕНТОВ: 1
МАКС ХАРАК-ТИКА: 16

Для продолжения нажмите любую клавишу . . .
```

Рисунок 4 – Работа оператора +

```

----- РАБОТА С ШАБЛОННЫМ СПИСКОМ -----
Какую команду вы хотите выполнить со списком?
1 | getLength() | узнать текущее количество элементов
2 | isExist(elem)) | узнать, есть ли элемент в списке
3 | insert(elem,n) | добавить элемент elem на позицию n
4 | remove(elem,n) | удалить элемент elem с позиции n
5 | + | добавить какой-либо элемент в начало списка
6 | - | удалить все вхождения какого-то элемента
7 | -- | удалить первый элемент из начала списка
8 | [] | узнать значени элемента по его индексу
9 | Вернуться к меню создания списка

Односвязный СПИСОК:
1 | строка
КОЛ-ВО ЭЛЕМЕНТОВ: 1
МАКС ХАРАК-ТИКА: 16

Введите номер команды: 3

elem = строка
n = 1

Односвязный СПИСОК:
1 | строка
2 | строка
КОЛ-ВО ЭЛЕМЕНТОВ: 2
МАКС ХАРАК-ТИКА: 16

Для продолжения нажмите любую клавишу . . .

```

Рисунок 5 – Работа метода insert

```

----- РАБОТА С ШАБЛОННЫМ СПИСКОМ -----
Какую команду вы хотите выполнить со списком?
1 | getLength() | узнать текущее количество элементов
2 | isExist(elem)) | узнать, есть ли элемент в списке
3 | insert(elem,n) | добавить элемент elem на позицию n
4 | remove(elem,n) | удалить элемент elem с позиции n
5 | + | добавить какой-либо элемент в начало списка
6 | - | удалить все вхождения какого-то элемента
7 | -- | удалить первый элемент из начала списка
8 | [] | узнать значени элемента по его индексу
9 | Вернуться к меню создания списка

Односвязный СПИСОК:
1 | строка
2 | строка
КОЛ-ВО ЭЛЕМЕНТОВ: 2
МАКС ХАРАК-ТИКА: 16

Введите номер команды: 6

elem = строка

Односвязный СПИСОК:
КОЛ-ВО ЭЛЕМЕНТОВ: 0
МАКС ХАРАК-ТИКА: 16

Для продолжения нажмите любую клавишу . . . ■

```

Рисунок 6 – Работа оператора -

```

----- РАБОТА С ШАБЛОННЫМ СПИСКОМ -----
Какую команду вы хотите выполнить со списком?
1 | getLength() | узнать текущее количество элементов
2 | isExist(elem)) | узнать, есть ли элемент в списке
3 | insert(elem,n) | добавить элемент elem на позицию n
4 | remove(elem,n) | удалить элемент elem с позиции n
5 | + | добавить какой-либо элемент в начало списка
6 | - | удалить все вхождения какого-то элемента
7 | -- | удалить первый элемент из начала списка
8 | [] | узнать значени элемента по его индексу
9 | Вернуться к меню создания списка

Односвязный СПИСОК:
КОЛ-ВО ЭЛЕМЕНТОВ: 0
МАКС ХАРАК-ТИКА: 16

Введите номер команды: 5

elem = строкаааа

Односвязный СПИСОК:
1 | строкаааа
КОЛ-ВО ЭЛЕМЕНТОВ: 1
МАКС ХАРАК-ТИКА: 16

Для продолжения нажмите любую клавишу . . .

```

Рисунок 7 – Работа оператора +

```

----- РАБОТА С ШАБЛОННЫМ СПИСКОМ -----
Какую команду вы хотите выполнить со списком?
1 | getLength() | узнать текущее количество элементов
2 | isExist(elem) | узнать, есть ли элемент в списке
3 | insert(elem,n) | добавить элемент elem на позицию n
4 | remove(elem,n) | удалить элемент elem с позиции n
5 | + | добавить какой-либо элемент в начало списка
6 | - | удалить все вхождения какого-то элемента
7 | -- | удалить первый элемент из начала списка
8 | [] | узнать значени элемента по его индексу
9 | Вернуться к меню создания списка

Односвязный СПИСОК:
1 | строкааа
КОЛ-ВО ЭЛЕМЕНТОВ: 1
МАКС ХАРАК-ТИКА: 16

Введите номер команды: 8
n = 1

Элемент № 1: строкааа

Для продолжения нажмите любую клавишу . . .

```

Рисунок 8 – Работа оператора []

```

----- РАБОТА С ШАБЛОННЫМ СПИСКОМ -----
Какую команду вы хотите выполнить со списком?
1 | getLength() | узнать текущее количество элементов
2 | isExist(elem) | узнать, есть ли элемент в списке
3 | insert(elem,n) | добавить элемент elem на позицию n
4 | remove(elem,n) | удалить элемент elem с позиции n
5 | + | добавить какой-либо элемент в начало списка
6 | - | удалить все вхождения какого-то элемента
7 | -- | удалить первый элемент из начала списка
8 | [] | узнать значени элемента по его индексу
9 | Вернуться к меню создания списка

Односвязный СПИСОК:
1 | строкааа
КОЛ-ВО ЭЛЕМЕНТОВ: 1
МАКС ХАРАК-ТИКА: 16

Введите номер команды: 5
elem = ещестрока

Односвязный СПИСОК:
1 | ещестрока
2 | строкааа
КОЛ-ВО ЭЛЕМЕНТОВ: 2
МАКС ХАРАК-ТИКА: 16

Для продолжения нажмите любую клавишу . . .

```

Рисунок 9 – Работа оператора +

```

----- РАБОТА С ШАБЛОННЫМ СПИСКОМ -----
Какую команду вы хотите выполнить со списком?
1 | getLength() | узнать текущее количество элементов
2 | isExist(elem) | узнать, есть ли элемент в списке
3 | insert(elem,n) | добавить элемент elem на позицию n
4 | remove(elem,n) | удалить элемент elem с позиции n
5 | + | добавить какой-либо элемент в начало списка
6 | - | удалить все вхождения какого-то элемента
7 | -- | удалить первый элемент из начала списка
8 | [] | узнать значени элемента по его индексу
9 | Вернуться к меню создания списка

Односвязный СПИСОК:
1 | ещестрока
2 | строкааа
КОЛ-ВО ЭЛЕМЕНТОВ: 2
МАКС ХАРАК-ТИКА: 16

Введите номер команды: 4
elem = ещестрока
n = 1

Односвязный СПИСОК:
1 | строкааа
КОЛ-ВО ЭЛЕМЕНТОВ: 1
МАКС ХАРАК-ТИКА: 16

Для продолжения нажмите любую клавишу . . .

```

Рисунок 10 – Работа метода remove


```

----- РАБОТА С ШАБЛОННЫМ СПИСКОМ -----
Какую команду вы хотите выполнить со списком?
1 | getLength() | узнать текущее количество элементов
2 | isExist(elem)) | узнать, есть ли элемент в списке
3 | insert(elem,n) | добавить элемент elem на позицию n
4 | remove(elem,n) | удалить элемент elem с позиции n
5 | + | добавить какой-либо элемент в начало списка
6 | - | удалить все вхождения какого-то элемента
7 | -- | удалить первый элемент из начала списка
8 | [] | узнать значени элемента по его индексу
9 | Вернуться к меню создания списка

Односвязный список:
1 | строкааа
КОЛ-ВО ЭЛЕМЕНТОВ: 1
МАКС ХАРАК-ТИКА: 16

Введите номер команды: 1

Кол-во элементов в списке: 1

Для продолжения нажмите любую клавишу . . .

```

Рисунок 11 – Работа метода getLength

```

----- РАБОТА С ШАБЛОННЫМ СПИСКОМ -----
Какую команду вы хотите выполнить со списком?
1 | getLength() | узнать текущее количество элементов
2 | isExist(elem)) | узнать, есть ли элемент в списке
3 | insert(elem,n) | добавить элемент elem на позицию n
4 | remove(elem,n) | удалить элемент elem с позиции n
5 | + | добавить какой-либо элемент в начало списка
6 | - | удалить все вхождения какого-то элемента
7 | -- | удалить первый элемент из начала списка
8 | [] | узнать значени элемента по его индексу
9 | Вернуться к меню создания списка

Односвязный список:
1 | строкааа
КОЛ-ВО ЭЛЕМЕНТОВ: 1
МАКС ХАРАК-ТИКА: 16

Введите номер команды: 2

elem = строкааа
да, такой элемент есть в списке

Для продолжения нажмите любую клавишу . . .

```

Рисунок 12 – Работа метода isExist

```

----- РАБОТА С ШАБЛОННЫМ СПИСКОМ -----
Какую команду вы хотите выполнить со списком?
1 | getLength() | узнать текущее количество элементов
2 | isExist(elem)) | узнать, есть ли элемент в списке
3 | insert(elem,n) | добавить элемент elem на позицию n
4 | remove(elem,n) | удалить элемент elem с позиции n
5 | + | добавить какой-либо элемент в начало списка
6 | - | удалить все вхождения какого-то элемента
7 | -- | удалить первый элемент из начала списка
8 | [] | узнать значени элемента по его индексу
9 | Вернуться к меню создания списка

Односвязный список:
1 | строкааа
КОЛ-ВО ЭЛЕМЕНТОВ: 1
МАКС ХАРАК-ТИКА: 16

Введите номер команды: 7

Односвязный список:
КОЛ-ВО ЭЛЕМЕНТОВ: 0
МАКС ХАРАК-ТИКА: 16

Для продолжения нажмите любую клавишу . . .

```

Рисунок 13 – Работа оператора --

Программа работает корректно, доступные пользователю команды работают без ошибок, некорректные действия пользователя и выходы из ОДЗ также учтены. Программа справляется с поставленной задачей.

