МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Отчет по выполнению практического задания 6

Тема:

Однонаправленный динамический список

Выполнил студент Цемкало А. Р.
Фамилия И. О.

группа ИКБО-10-20

Москва 2021

СОДЕРЖАНИЕ

Цель. Получить знания и практические навыки управления динамическим однонаправленным списком.

Вариант 10.

1. Постановка задачи.

Реализуйте программу решения задачи варианта по использованию линейного однонаправленного списка.

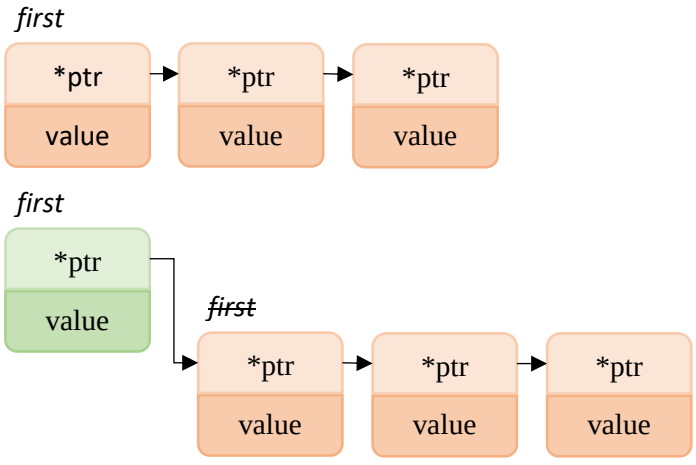
2. Определение списка операций над списком, которые выявлены в процессе исследования задач дополнительного задания.

2.1. Определить структуру узла однонаправленного списка в соответствии с вариантом.

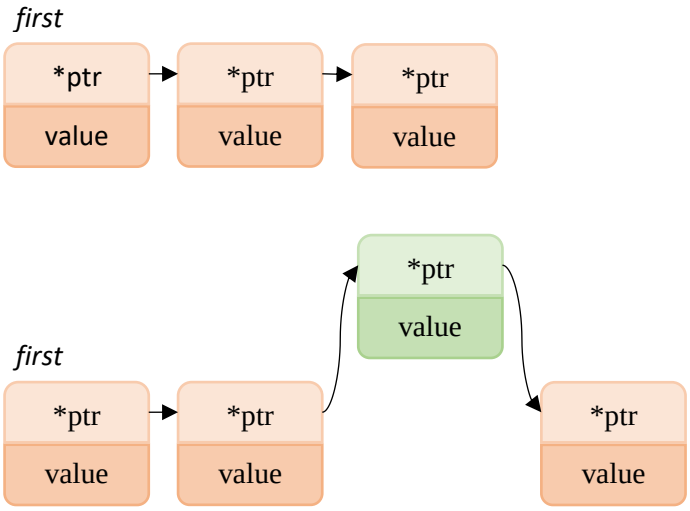
```
struct Node {
	char val;
	Node* ptr;
	Node(char _val) : ptr(nullptr), val(_val) {}
};
```

2.2. Изобразить (рисунок) для каждой операции полученного списка процесс выполнения операции на существующем однонаправленном списке.

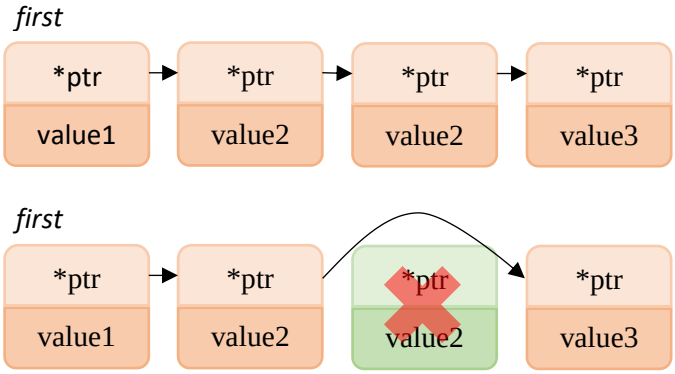
Функция вставки нового узла перед первым узлом



Функция вставки нового узла после конкретного

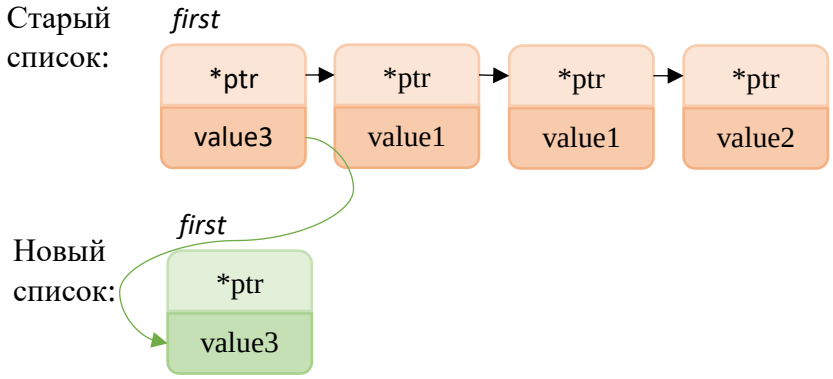


Функция, которая в каждой последовательности одинаковых символов оставляет только один

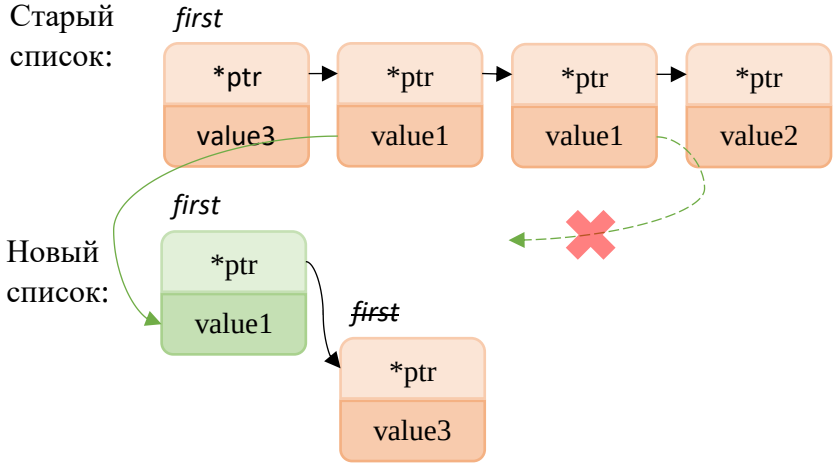


Функция, которая создает новый список из цифр исходного, выполняя вставку элемента в новый список в порядке возрастания цифр

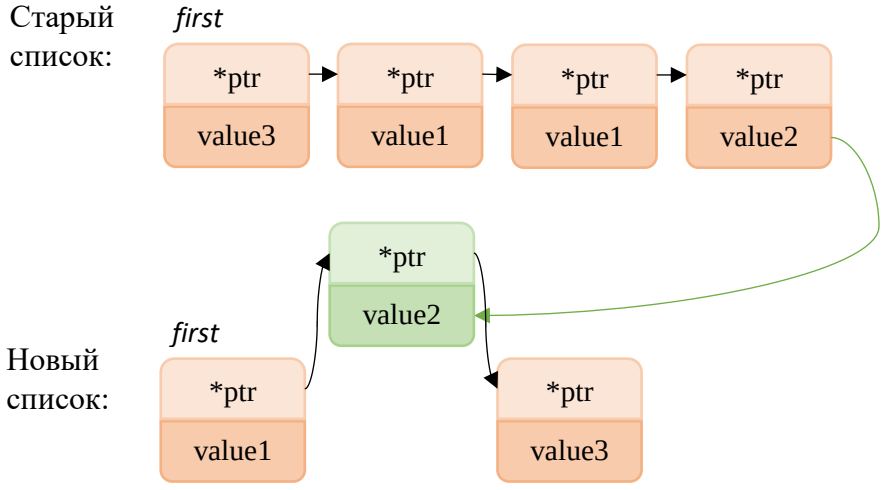
1.



2.



3.



2.3. Изобразите структуру данных, которая будет использоваться в операциях.

```
struct list {
	Node* first;
	Node* last;

	list() : first(nullptr), last(nullptr) {}

	bool is_empty();
	Node* add(char value, Node* current = NULL);
	void print();
	char longest_sequence();
	Node* find_previous(Node* p);
	void remove_same_symbols();
	bool symbol_in_list(char symb);
	list make_sorted_list();
};
```

2.4. Привести алгоритм выполнения операции

Функция вставки нового узла:

Если список пустой, создаём первый узел. Прекращение работы функции.

Если в аргументах был указан текущий узел, после которого вставляется новый узел: в указатель нового узла записываем указатель на следующий узел, на который ранее был у текущего, а текущему узлу указываем на новый.

Функция, которая определяет в списке L самую длинную последовательность, состоящую из одинаковых символов:

В переменную symb_1 типа char записывается значение предыдущего узла, в symb_2 типа char записывается значение нового узла, в переменную k записывается количество совпадений symb_1 и symb_2, если k больше k_max (текущего максимального количества повторяющихся узлов), в symb_max записывается значение такого узла.

Функция поиска предыдущего узла: здесь идёт поиск узла, который указывает на нужный нам.

Функция, которая в каждой последовательности одинаковых символов оставляет только один:

Если значения узла повторяются, оставляем только один, остальные удаляем, записав в указатель предыдущего указатель удаляемого (т.е. указатель на новый узел). Текущим узлом становится узел, стоявший до удалённого узла.

Функция, которая создаёт новый список из цифр исходного, выполняя вставку элемента в новый список в порядке возрастания цифр:

В новый список добавляется первый узел старого списка. В цикле берётся элемент из старого списка и вставляется в нужное место нового, так повторяется пока не цикл не дойдёт до конца старого списка.

2.5. Привести таблицу тестов для тестирования каждой операции

Функция вставки нового узла

Номер теста	Входные данные	Ожидаемый список	Список после выполнения программы
1	Вставка в начало списка. Добавляем поочерёдно ‘k’, ‘l’, ‘c’.	c l k	c l k
2	Вставить ‘b’, после ‘b’ вставить ‘c’, в начало ‘a’.	a b c	a b c

Функция, которая определяет в списке L самую длинную последовательность, состоящую из одинаковых символов

Номер теста	Входные данные	Ожидаемый результат	Результат выполнения программы
1	x a a a a h h h c c c l k	a	a
2	x c c c a a a a h h h h h c c c l k	h	h

Функция, которая в каждой последовательности одинаковых символов оставляет только один

Номер теста	Входные данные	Ожидаемый результат	Результат выполнения программы
1	x c c a a a a h c c c l k	x c a h c l k	x c a h c l k
2	x c a h c l k	x c a h c l k	x c a h c l k

Функция, которая создает новый список из цифр исходного, выполняя вставку элемента в новый список в порядке возрастания цифр

Номер теста	Входные данные	Ожидаемый результат	Результат выполнения программы
1	e c f b a a d	a b c d e f	a b c d e f
2	x c c a a a a h c c c l k	a c h k l x	a c h k l x

3. Код программы

```
#include <iostream>
using namespace std;

struct Node {
	char val;
	Node* ptr;
	Node(char value) : ptr(nullptr), val(value) {}
};

struct list {
	Node* first;
	Node* last;

	list() : first(nullptr), last(nullptr) {}

	bool is_empty() {
		return first == nullptr && last == nullptr;
	}

	Node* add(char value, Node* current = NULL) {
		Node* p = new Node(value);
		if (is_empty()) {
			first = p;
			last = p;
			return p;
		}
		if (current == NULL) {
```

```
                p->ptr = first;
                first = p;
        }
        else {
                p->ptr = current->ptr;
                current->ptr = p;
        }
        return p;
}

void print() {
        if (is_empty()) return;
        Node* p = first;
        while (p) {
                cout << p->val << " ";
                p = p->ptr;
        }
}

char longest_sequence() {
        if (!is_empty()) {
                char symb_1, symb_2, symb_max;
                int k = 0, k_max = 1;
                Node* p = first;
                symb_max = p->val;
                symb_1 = p->val;
                while (p) {
                        symb_2 = p->val;
                        if (symb_1 == symb_2) {
                                k++;
                        }
                        else {
                                symb_1 = symb_2;
                                k = 1;
                        }
                        if (k > k_max) {
                                symb_max = symb_1;
                                k_max = k;
                        }
                        p = p->ptr;
                }
                return symb_max;
        }
}
Node* find_previous(Node* p) {
        if (p == first) {
                first = p->ptr;
                return p;
        }
        Node* previous_p = first;
        while (previous_p->ptr != p) {
                previous_p = previous_p->ptr;
        }
        return previous_p;
}

void remove_same_symbols() {
        if (is_empty()) return;
        char symb_1, symb_2;
        Node* p = first;
        symb_1 = ' ';
        while (p) {
                symb_2 = p->val;
                if (symb_1 == symb_2) {
                        Node* previous_p = find_previous(p);
```

```
                        previous_p->ptr = p->ptr;
                        delete p;
                        p = previous_p;
                }
                else symb_1 = symb_2;
                p = p->ptr;
        }
    }

    bool symbol_in_list(char symb) {
        Node* p = first;
        while (p) {
            if (symb == p->val) return true;
            p = p->ptr;
        }
        return false;
    }

    list make_sorted_list() {
        list new_list;
        if (is_empty()) return new_list;
        Node* p = first;
        while (p) {
            if (!new_list.symbol_in_list(p->val)) {
                if (new_list.is_empty()) new_list.add(p->val);
                else {
                    Node* new_p = new_list.first;
                    while (new_p->val < p->val && new_p != new_list.last) {
                        new_p = new_p->ptr;
                    }
                    if (new_p->val > p->val) {
                        if (new_p->ptr == new_list.first->ptr)
new_list.add(p->val);
                        else new_list.add(p->val,
new_list.find_previous(new_p));
                    }
                    else new_list.add(p->val, new_p);

                }
            }
            p = p->ptr;
        }
        return new_list;
    }
};

int main() {
    list L;
    cout << "The list is created" << endl;
    if (L.is_empty()) {
        cout << "The list is empty" << endl;
    }
    else {
        cout << "The list is not empty" << endl;
    }

    L.add('k');
    L.add('l');
    L.add('c');
    L.add('c');
    L.add('c');
    L.add('h');
    L.add('a');
    L.add('a');
    L.add('a');
```

```
	L.add('a');
	L.add('c');
	L.add('c');
	L.add('x');

	cout << "Some symbols are added to the list" << endl << "This is our list: ";
	L.print();
	cout << endl;
	if (L.is_empty()) {
		cout << "The list is empty" << endl;
	}
	else {
		cout << "The list is not empty" << endl;
	}
	cout << "The symbol repeated more times in a row is: " << L.longest_sequence() <<
endl;

	cout << "Let's remove repeated symbols." << endl << "Now our list is: ";
	L.remove_same_symbols();
	L.print();

	cout << "New sorted list made of our old list without taking repeating symbols is: ";
	list new_L = L.make_sorted_list();
	new_L.print();

	return 0;
}
```

4. Результат тестирования программы: скриншоты выполнения каждой операции.

Функция вставки нового узла

```
int main() {
    list L;
    cout << "The list is created" << endl;
    if (L.is_empty()) {
        cout << "The list is empty" << endl;
    }
    else {
        cout << "The list is not empty" << endl;
    }

    Node* p = L.add('b');
    L.add('c', p);
    L.add('a');
    //L.add('k');
    //L.add('l');
    //L.add('c');
    //L.add('c');
    //L.add('c');
    //L.add('h');
    //L.add('a');
    //L.add('a');
    //L.add('a');
    //L.add('a');
    //L.add('c');
    //L.add('c');
    //L.add('x');

    cout << "Some symbols are added to the list" << en
    L.print();
    cout << endl;
```

```
The list is created
The list is empty
Some symbols are added to the list
This is our list: a b c
```

Функция вставки нового узла перед первым узлом

```
int main() {
    list L;
    cout << "The list is created" << endl;
    if (L.is_empty()) {
        cout << "The list is empty" << endl;
    }
    else {
        cout << "The list is not empty" << endl;
    }

    //Node* p = L.add('b');
    //L.add('c', p);
    L.add('a');
    L.add('k');
    L.add('l');
    L.add('c');
    L.add('c');
    L.add('c');
    L.add('h');
    L.add('a');
    L.add('a');
    L.add('a');
    L.add('a');
    L.add('c');
    L.add('c');
    L.add('x');

    cout << "Some symbols are added to the list" <<
    L.print();
    cout << endl;
```

```
The list is created
The list is empty
Some symbols are added to the list
This is our list: x c c a a a a h c c c l k a
```

Функция, которая определяет в списке L самую длинную последовательность, состоящую из одинаковых символов

```
The list is created
Some symbols are added to the list
This is our list: x c c a a a a h c c c l k a
The symbol repeated more times in a row is: a
```

Функция, которая в каждой последовательности одинаковых символов оставляет только один

```
The list is created
Some symbols are added to the list
This is our list: x c c a a a a h c c c l k a
Let's remove repeated symbols.
Now our list is: x c a h c l k a
```

Функция, которая создает новый список из цифр исходного, выполняя вставку элемента в новый список в порядке возрастания цифр (В новом списке не может быть повторяющихся цифр)

```
The list is created
Some symbols are added to the list
This is our list: x c c a a a a h c c c l k a
New sorted list made of our old list without taking repeating symbols is: a c h k l x
```

ВЫВОДЫ

В ходе выполнения задания получены знания и практические навыки управления динамическим однонаправленным списком. Разработаны:

1. Функция для создания исходного списка, используя функцию вставки нового узла перед первым узлом;
2. Функция вывода списка;
3. Функция, которая определяет в списке L самую длинную последовательность, состоящую из одинаковых символов;
4. Функция, которая в каждой последовательности одинаковых символов оставляет только один;
5. Функция, которая создает новый список из цифр исходного, выполняя вставку элемента в новый список в порядке возрастания цифр.

Тестирования всех операций пройдены успешно.

СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

1. Процедурное программирование Языки программирования – Сайт lizochekk! [Электронный ресурс]: URL: https://lizochekk.jimdofree.com/программирование/
2. Документация по Microsoft C/C++ | Microsoft Docs – [Электронный ресурс] URL: https://docs.microsoft.com/ru-ru/cpp/?view=msvc-160
3. Все публикации подряд / Хабр – [Электронный ресурс] URL: https://habr.com/ru/