Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Пермский национальный исследовательский**

**политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т**

**по лабораторной работе №11.2**

Дисциплина: основы алгоритмизации и программирования

Тема: “Двунаправленный список"

**Вар.20**

Выполнил работу

студент группы ИВТ-20-2Б

Кузнецов Н.Д

Проверил

Доцент кафедры ИТАС

Полякова О.А.

Пермь, 2020

**Цель задачи**

1) Получить практические навыки работы с однонаправленными списками;

2) Получить практические навыки работы с двунаправленными списками;

3) Получить практические навыки работы со стеками;

4) Получить практические навыки работы с очередями.  
Для достижения поставленной цели необходимо решить следующие задачи:

* Разработать структуру однонаправленного списка.
* Организовать ввод-вывод данных.
* Реализовать функцию удаления элементов из списка .
* Разработать программу.

**Постановка задачи**

Реализовать двунаправленный список. Тип информационного поля char\*. Добавить в список элемент после элемента с заданным информационным полем.

**Анализ задачи**

1. Для решения задачи используются типы данных **bool, int** и класс **string**.
2. Для хранения вводимых данных реализован двунаправленный список с полем типа string.   
   struct List

{

string data;

List\* next\_block;

List\* previous\_block;

};

1. Для заполнения списка реализована функция CreateList, которая принимает в качестве параметра размер списка и возвращает указатель на первый элемент списка.  
   List\* CreateList(int size)

{

List\* first\_element = nullptr;

List\* current\_element;

if (size == 1)

{

cout << "Введите 0-й элемент списка:\t";

current\_element = new List;

cin >> current\_element->data;

first\_element = current\_element;

current\_element->next\_block = nullptr;

current\_element->previous\_block = nullptr;

}

else

{

cout << "Введите 0-й элемент списка:\t";

current\_element = new List;

cin >> current\_element->data;

first\_element = current\_element;

current\_element->next\_block = nullptr;

current\_element->previous\_block = nullptr;

for (int i = 1; i < size; i++)

{

cout << "Введите " << i << "-й элемент списка:\t";

List\* new\_element = new List;

current\_element->next\_block = new\_element;

new\_element->previous\_block = current\_element;

current\_element = new\_element;

cin >> current\_element->data;

current\_element->next\_block = nullptr;

}

}

return first\_element;

}  
Для вывода списка реализована функция PrintList, в качестве параметра принимает указатель на 1 элемент списка, работает до тех пор, пока указатель на текущий элемент не равен **nullptr**.

void PrintList(List\* first\_element)

{

if (first\_element == nullptr)

{

cout << "Список пуст!" << endl;

return;

}

int size = Size(first\_element);

cout << "Вывод списка в прямом порядке:" << endl;

int i = 0;

while (first\_element != nullptr)

{

cout << i << "-й элемент массива:\t" << first\_element->data << endl;

first\_element = first\_element->next\_block;

i++;

}

cout << endl;

}

1. Описывается структура List с 3 полями (int data, List\* next\_block, List\* previous\_block).
2. В функциях ввода-вывода используются операторы cin, cout (см.п.3).
3. В функции AddElement описывается переменная типа string (value), ее значение вводит пользователь. Так же создается переменная типа bool (check), если в списке найдется поле удовлетворяющее value, то переменная check поменяет свое значение на true, в противном случае, она будет хранить в себе значение false. Описывается указатель на первый элемент списка (List\* current\_element = first\_element), этот указатель будет использоваться для получения значения текущего элемента списка. Добавление элемента в список осуществляется через цикл while, цикл выполняется до тех пор, пока текущий элемент не указывает на nullptr. Внутри цикла рассмотрено 2 случая:  
   **А)** Текущий элемент удовлетворяет value и следующий элемент списка существует.  
   **Б)** Текущий элемент удовлетворяет value и следующего элемента в списке нет.  
   if (current\_element->data == value && current\_element->next\_block != nullptr)

{

List\* tmp1 = new List;

cout << "Введите значение нового элемента:\t";

cin >> tmp1->data;

List\* tmp2 = current\_element->next\_block;

tmp2->next\_block = current\_element->next\_block->next\_block;

current\_element->next\_block = tmp1;

tmp1->previous\_block = current\_element;

tmp1->next\_block = tmp2;

tmp2->previous\_block = tmp1;

check = true;

}

else if (current\_element->data == value && current\_element->next\_block == nullptr)

{

List\* tmp1 = new List;

cout << "Введите значение нового элемента:\t";

cin >> tmp1->data;

current\_element->next\_block = tmp1;

tmp1->previous\_block = current\_element;

tmp1->next\_block = nullptr;

check = true;

}  
**Случай** **А**: для добавления элемента в список описывается новый указатель, ему присваивается адрес выделенной памяти из кучи. Пользователь вводит значение в поле этого элемента. Создается промежуточный указатель, ему присваивается адрес следующего элемента в списке. Далее происходит обычная вставка в список.  
…   
List\* tmp2 = current\_element->next\_block;

tmp2->next\_block = current\_element->next\_block->next\_block;

current\_element->next\_block = tmp1;

tmp1->previous\_block = current\_element;

tmp1->next\_block = tmp2;

tmp2->previous\_block = tmp1;  
…   
  
**Случай Б:** аналогично случаю А создается новый элемент списка, который инициализирует пользователь. Полю next\_block текущего элемента присваивается значение нового указателя, полю previous\_block нового элемента присваивается значение текущего элемента, полю next\_block нового элемента присваивается значение nullptr.  
…   
current\_element->next\_block = tmp1;

tmp1->previous\_block = current\_element;

tmp1->next\_block = nullptr;  
…   
После всех возможных изменений в списке функция возвращает значение переменной check.

**Код через struct**

#include <iostream>

#include <string>

#include <vector>

using namespace std;

struct List

{

string data;

List\* next\_block;

List\* previous\_block;

};

int GetSize()

{

int size;

cin >> size;

while (size <= 0)

{

cout << "Введите корректное число:\t";

cin >> size;

}

return size;

}

List\* CreateList(int size)

{

List\* first\_element = nullptr;

List\* current\_element;

if (size == 1)

{

cout << "Введите 0-й элемент списка:\t";

current\_element = new List;

cin >> current\_element->data;

first\_element = current\_element;

current\_element->next\_block = nullptr;

current\_element->previous\_block = nullptr;

}

else

{

cout << "Введите 0-й элемент списка:\t";

current\_element = new List;

cin >> current\_element->data;

first\_element = current\_element;

current\_element->next\_block = nullptr;

current\_element->previous\_block = nullptr;

for (int i = 1; i < size; i++)

{

cout << "Введите " << i << "-й элемент списка:\t";

List\* new\_element = new List;

current\_element->next\_block = new\_element;

new\_element->previous\_block = current\_element;

current\_element = new\_element;

cin >> current\_element->data;

current\_element->next\_block = nullptr;

}

}

return first\_element;

}

void PrintList(List\* first\_element)

{

if (first\_element == nullptr)

{

cout << "Список пуст!" << endl;

return;

}

cout << "Вывод списка:" << endl;

int i = 0;

while (first\_element != nullptr)

{

cout << i << "-й элемент массива:\t" << first\_element->data << endl;

first\_element = first\_element->next\_block;

i++;

}

cout << endl;

}

bool AddElement(List\*& first\_element)

{

string value;

cout << "Введите поле элемента, после которого нужно вставить новый элемент:\t";

cin >> value;

bool check = false;

List\* current\_element = first\_element;

while (current\_element != nullptr)

{

if (current\_element->data == value && current\_element->next\_block != nullptr)

{

List\* tmp1 = new List;

cout << "Введите значение нового элемента:\t";

cin >> tmp1->data;

List\* tmp2 = current\_element->next\_block;

tmp2->next\_block = current\_element->next\_block->next\_block;

current\_element->next\_block = tmp1;

tmp1->previous\_block = current\_element;

tmp1->next\_block = tmp2;

tmp2->previous\_block = tmp1;

check = true;

}

else if (current\_element->data == value && current\_element->next\_block == nullptr)

{

List\* tmp1 = new List;

cout << "Введите значение нового элемента:\t";

cin >> tmp1->data;

current\_element->next\_block = tmp1;

tmp1->previous\_block = current\_element;

tmp1->next\_block = nullptr;

check = true;

}

current\_element = current\_element->next\_block;

}

return check;

}

int main()

{

setlocale(LC\_ALL, "ru");

List\* list;

cout << "Введите длину списка:\t";

int size = GetSize();

cout << "=====================================================" << endl;

list = CreateList(size);

cout << "=====================================================" << endl;

cout << "Изначальный список:" << endl << endl;

PrintList(list);

cout << "=====================================================" << endl;

if (AddElement(list))

{

cout << "=====================================================" << endl;

cout << "Измененный список:" << endl << endl;

PrintList(list);

cout << "=====================================================" << endl;

}

else

{

cout << "Такого элемента нет!" << endl;

}

}

**Код через STL**

#include <iostream>

#include <list>

#include <string>

#include <iterator>

using namespace std;

list <string> list\_of\_char;

void Create\_list(int size)

{

string tmp;

for (int i = 0; i < size; i++)

{

cout << "Введите " << i << "-й элемент списка:\t";

cin >> tmp;

list\_of\_char.push\_back(tmp);

}

}

void Print\_list()

{

list <string> ::iterator startpos = list\_of\_char.begin();

list <string> ::iterator endpos = list\_of\_char.end();

int i = 0;

for (startpos; startpos != endpos; startpos++)

{

cout << i << "-й элемент списка:\t" << \*startpos << endl;

i++;

}

}

bool AddElement()

{

string value;

bool check = false;

list <string> ::iterator startpos = list\_of\_char.begin();

list <string> ::iterator endpos = list\_of\_char.end();

cout << "Введите имя поля, после которого нужно добавить элемент:\t";

cin >> value;

for (startpos; startpos != endpos; startpos++)

{

if ((\*startpos) == value)

{

string tmp;

list <string> ::iterator currentpos = startpos;

cout << "Введите значение нового поля:\t";

cin >> tmp;

list\_of\_char.insert(++currentpos, tmp);

check = true;

}

}

return check;

}

int GetSize()

{

int size;

cout << "Введите размер списка:\t";

cin >> size;

while (size < 1)

{

cout << "Введите корректное число:\t";

cin >> size;

}

return size;

}

int main()

{

setlocale(LC\_ALL, "ru");

int size;

size = GetSize();

Create\_list(size);

cout << "=============================" << endl;

cout << "Список перед изменением:" << endl;

Print\_list();

cout << "=============================" << endl;

if (AddElement())

{

cout << "=============================" << endl;

cout << "Список после изменения:" << endl;

Print\_list();

cout << "=============================" << endl;

}

else

{

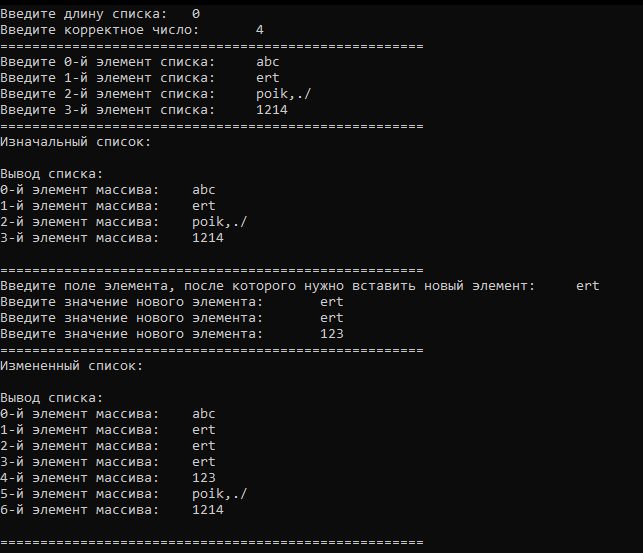
cout << "Такого элемента в списке нет!" << endl;

cout << "=============================" << endl;

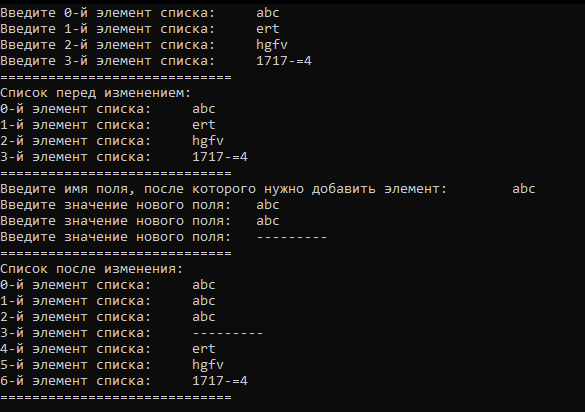
}

}

**Работа кода через struct**



**Работа код через STL**



**Блок-схема**

