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**

Дисциплина: основы алгоритмизации и программирования

Тема: “Динамические структуры данных"

**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 однонаправленный список. Тип информационного поля int. Удалить из списка последний элемент с четным информационным полем.

**Анализ задачи**

1. Для решения задачи используются типы данных **bool, int**.
2. Для хранения вводимых данных реализован однонаправленный список с полем типа int. Так же для хранения индексов четных элементов в списке используется класс **vector**.  
   #include <vector>  
   struct List

{

int data;

List\* next\_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:\t";

current\_element = new List;

cin >> (\*current\_element).data;

first\_element = current\_element;

current\_element->next\_block = nullptr;

}

else

{

cout << "Введите 0-й элемент:\t";

current\_element = new List;

cin >> (\*current\_element).data;

first\_element = current\_element;

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

{

List\* new\_element = new List;

(\*current\_element).next\_block = new\_element;

current\_element = new\_element;

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

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\* current\_element)

{

if (current\_element == nullptr)

{

cout << "Список пуст!" << endl;

return;

}

int i = 0;

while (current\_element != nullptr)

{

cout << i << "-й элемент списка\t" << (\*current\_element).data << endl;

current\_element = (\*current\_element).next\_block;

i++;

}

}

1. Описывается структура List с 2 полями (int data, List\* next\_block).  
   struct List

{

int data;

List\* next\_block;

};

1. В функциях ввода-вывода (см.п.3) используются операторы cin, cout.
2. В функции DeleteElement используется vector для запоминания индексов элементов списка, которые удовлетворяют удалению. В качестве параметра функция принимает ссылку на указатель первого элемента списка. Первым циклом for реализована проверка элементов списка на удаление. Если же поле элемента списка делится нацело на 2, то в vector добавляется индекс этого элемента (в данном случае первый элемент списка имеет индекс 0). Далее оператором ветвления if проверяется размер vector’а, если его размер равен нулю, то пользователю выводится сообщение о том, что в списке нет четных элементов. Если же таким числом является первый элемент списка, то он удаляется, указателю на первый элемент присваивается значение указателя следующего элемента списка. В противном случае функция проходится циклом for по всем элементам списка, если же индекс списка удовлетворяет последнему значению в векторе - 1, то происходит удаление этого элемента. Для удаления создается промежуточный указатель, которому присваивается значение следующего элемента списка. Полю next\_block текущего элемента присваивается значение указателя элемента, через 1 в списке, удаляется освобождается выделенная память под промежуточный указатель на элемент списка.  
   bool DeleteElement(List\*& first\_element)

{

vector <int> index;

int size = Size(first\_element);

List\* current\_element = first\_element;

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

{

if (current\_element->data % 2 == 0)

{

index.push\_back(i);

}

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

}

current\_element = first\_element;

if (index.size() == 0)

{

cout << "Четных элементов нет!" << endl;

return false;

}

if (index[index.size() - 1] == 0)

{

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

delete current\_element;

}

else

{

for (int i = 0; i < size - 1; i++)

{

if (i == index[index.size() - 1] - 1) //предыдущий блок от того элемента, который мы хотим удалить

{

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

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

delete tmp;

}

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

}

} return true;}

**Код**

#include <iostream>

#include <vector>

using namespace std;

struct List

{

int data;

List\* next\_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:\t";

current\_element = new List;

cin >> (\*current\_element).data;

first\_element = current\_element;

current\_element->next\_block = nullptr;

}

else

{

cout << "Введите 0-й элемент:\t";

current\_element = new List;

cin >> (\*current\_element).data;

first\_element = current\_element;

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

{

List\* new\_element = new List;

(\*current\_element).next\_block = new\_element;

current\_element = new\_element;

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

cin >> (\*current\_element).data;

(\*current\_element).next\_block = nullptr;

}

}

return first\_element;

}

void PrintList(List\* current\_element)

{

if (current\_element == nullptr)

{

cout << "Список пуст!" << endl;

return;

}

int i = 0;

while (current\_element != nullptr)

{

cout << i << "-й элемент списка\t" << (\*current\_element).data << endl;

current\_element = (\*current\_element).next\_block;

i++;

}

}

int Size(List\* current\_element)

{

int k = 0;

while (current\_element != nullptr)

{

current\_element = (\*current\_element).next\_block;

k++;

}

return k;

}

bool DeleteElement(List\*& first\_element)

{

vector <int> index;

int size = Size(first\_element);

List\* current\_element = first\_element;

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

{

if (current\_element->data % 2 == 0)

{

index.push\_back(i);

}

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

}

current\_element = first\_element;

if (index.size() == 0)

{

cout << "Четных элементов нет!" << endl;

return false;

}

if (index[index.size() - 1] == 0)

{

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

delete current\_element;

}

else

{

for (int i = 0; i < size - 1; i++)

{

if (i == index[index.size() - 1] - 1) //предыдущий блок от того элемента, который мы хотим удалить

{

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

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

delete tmp;

}

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

}

}

return true;

}

int main()

{

setlocale(LC\_ALL, "ru");

cout << "Введите размер списка:\t";

int size = GetSize();

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

List\* List\_of\_int\_elements = CreateList(size);

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

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

PrintList(List\_of\_int\_elements);

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

bool check = DeleteElement(List\_of\_int\_elements);

if (!check) return 0;

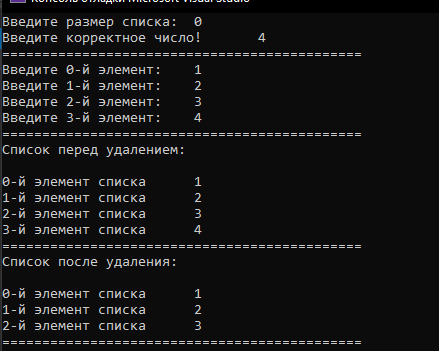
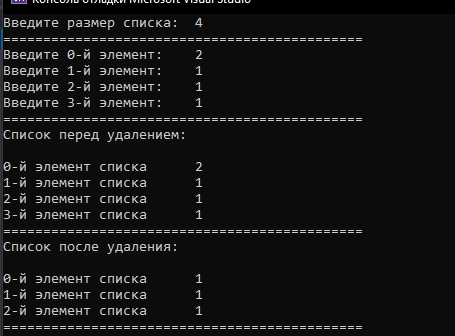
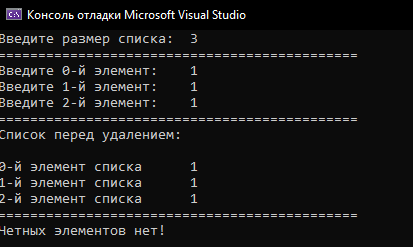
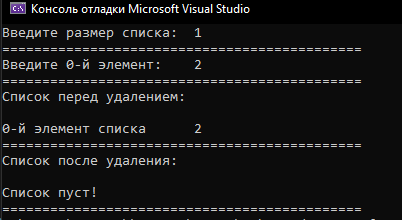
cout << "Список после удаления:" << endl << endl;

PrintList(List\_of\_int\_elements);

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

}

**Работа кода**

**Блок-схема**

