Пермский Национальный Исследовательский Политехнический Университет

Лабораторная работа №11.1

"Динамические структуры данных"

25 вариант

Выполнил:

Студент группы ИВТ-20-2б

Карелов Вадим Андреевич

Проверила доцент кафедры ИТАС

Полякова О.А.

Пермь 2021

**Постановка задачи**

1. Сформировать однонаправленный и двунаправленный списки или стек и очередь. Тип информационного поля указан в варианте. (Тип информационного поля int. Удалить из списка все элементы с четными информационными полями.)

2. Распечатать полученную структуру.

3. Выполнить обработку структуры в соответствии с заданием.

4. Распечатать полученный результат.

**Анализ задачи**

1. *Используемые типы данных.*

В программе используются: структура List, int.

Структура List используется для хранения одного элемента типа int.

List\* first, \* p;

Тип int используется для хранения данных, как параметр в цикле, для временного хранения размера динамической структуры.

int size;

1. *Действия над используемыми данными.*

С данными типа int производятся следующие действия: математические операции (инкремент в циклах с параметром).

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

С данными типа List производятся следующие действия: создание односвязного списка, добавление элементов, удаление элементов, вывод на экран. (Код представлен в пункте e)

1. *Вид данных.*

Односвязный список представлен в программе как структура, одной из полей которой указывает на следующий элемент списка – такая же структура. (Код в пункте г)

1. *Структура.*

В программе используется структура List, которая реализует односвязный список. Поле data имеет тип int и предназначено для хранения значения. Поле next имеет тип List\* и предназначено для хранения указателя на следующий элемент списка.

struct List

{

int data;

List\* next;

};

1. *Ввод и вывод.*

Ввод и вывод всех данных осуществляется в консоль через операторы cin и cout соответственно.

cout << "Заполнение списка на " << size << " элементов" << endl;

cin >> p->data;

1. *Действия для решения задачи.*

Для создания списка используется цикл с параметром и оператор ветвления. Функция возвращает указатель на первый элемент.

List\* CreateList(int size)

{

cout << "Заполнение списка на " << size << " элементов" << endl;

if (size <= 0)

{

cout << "Список не может быть создан" << endl;

return 0;

}

List\* first, \* p;

first = NULL;

cout << "Введите элемент 1: ";

p = new List;

cin >> p->data;

first = p;

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

{

List\* h = new List;

p->next = h;

p = p->next;

cout << "Введите элемент " << i + 1 << ": ";

cin >> p->data;

p->next = NULL;

}

return first;

}

Для печати списка используется оператор ветвления и цикл с предусловием.

void PrintList(List\* first)

{

cout << "Вывод списка" << endl;

if (first == NULL)

cout << "Список пустой" << endl;

else

{

List\* p = first;

while (p != NULL)

{

cout << p->data << " ";

p = p->next;

}

cout << endl;

}

}

Для удаления списка используется цикл с предусловием и операторы ветвления.

void DeleteElements(List\* &first)

{

List\* newFirst = first;

while (first != NULL && first->data % 2 == 0)

{

newFirst = first->next;

delete first;

first = newFirst;

}

while (first != NULL && first->next != NULL)

{

if (first->next->data % 2 == 0)

{

List\* newNextElement = first->next->next;

delete first->next;

first->next = newNextElement;

}

else

{

first = first->next;

}

}

first = newFirst;

}

Для ввода корректного числа используется цикл с постусловием.

int size;

do

{

cout << "Введите количество элементов списка" << endl;

cin >> size;

} while (size < 1);

**Код программы**

#include <iostream>

using namespace std;

struct List

{

int data;

List\* next;

};

List\* CreateList(int size)

{

cout << "Заполнение списка на " << size << " элементов" << endl;

if (size <= 0)

{

cout << "Список не может быть создан" << endl;

return 0;

}

List\* first, \* p;

first = NULL;

cout << "Введите элемент 1: ";

p = new List;

cin >> p->data;

first = p;

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

{

List\* h = new List;

p->next = h;

p = p->next;

cout << "Введите элемент " << i + 1 << ": ";

cin >> p->data;

p->next = NULL;

}

return first;

}

void PrintList(List\* first)

{

cout << "Вывод списка" << endl;

if (first == NULL)

cout << "Список пустой" << endl;

else

{

List\* p = first;

while (p != NULL)

{

cout << p->data << " ";

p = p->next;

}

cout << endl;

}

}

void DeleteElements(List\* &first)

{

List\* newFirst = first;

while (first != NULL && first->data % 2 == 0)

{

newFirst = first->next;

delete first;

first = newFirst;

}

while (first != NULL && first->next != NULL)

{

if (first->next->data % 2 == 0)

{

List\* newNextElement = first->next->next;

delete first->next;

first->next = newNextElement;

}

else

{

first = first->next;

}

}

first = newFirst;

}

int main()

{

setlocale(LC\_ALL, "Russian");

int size;

do

{

cout << "Введите количество элементов списка" << endl;

cin >> size;

} while (size < 1);

List \*list = CreateList(size);

PrintList(list);

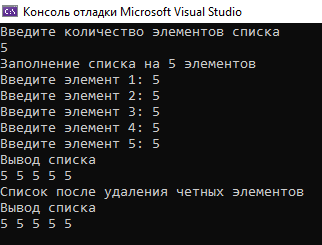
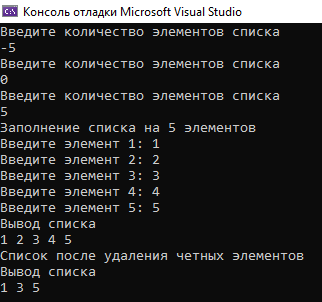
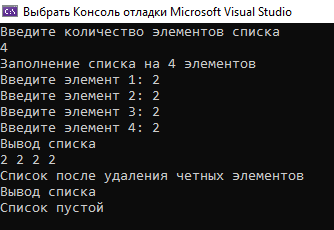
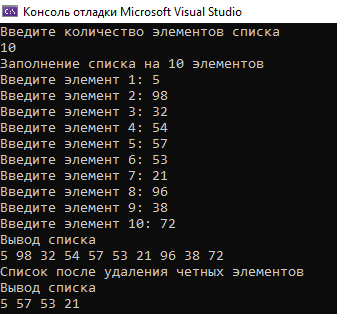
DeleteElements(list);

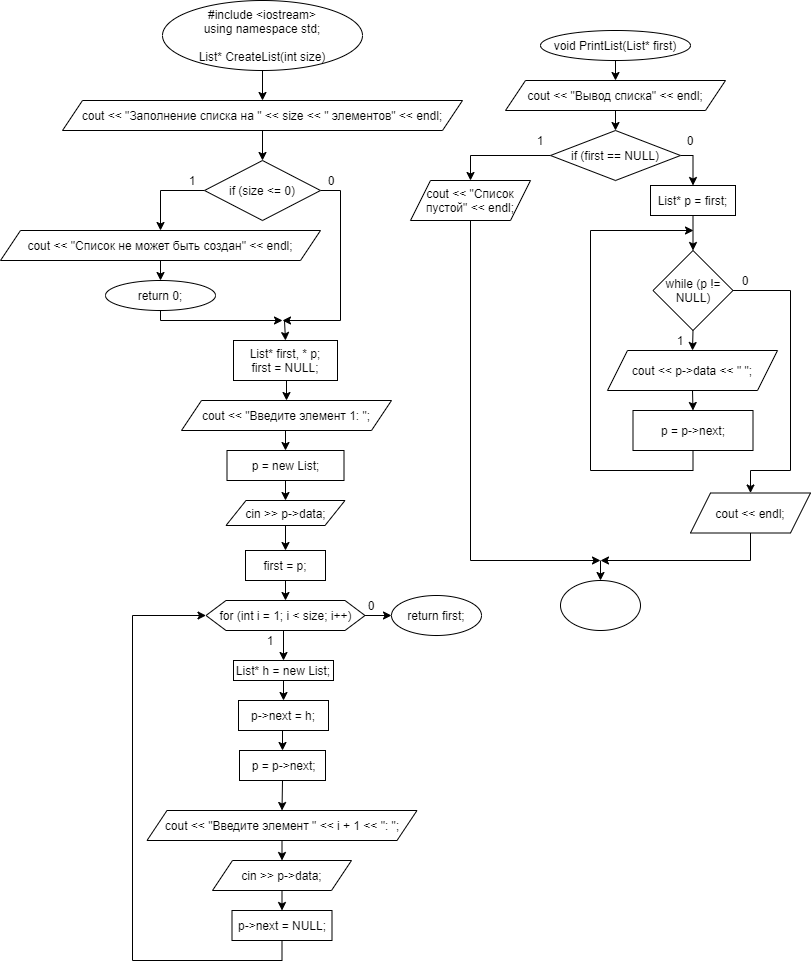
cout << "Список после удаления четных элементов" << endl;

PrintList(list);

}

**Скриншоты**

**  **

**Блок-схема**

