Пермский Национальный Исследовательский Политехнический Университет

Лабораторная работа №11.2

"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, list<int>.

1.1. Структура List используется для хранения одного элемента типа int.

spisok\* p = new spisok;

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

Int size;

1.3. Тип int используется для хранения данных внутри структуры.

cin>> p->data;

1.4. Тип list<int>используется для хранения двусвязного списка через STL.

1. *Действия над используемыми данными.*

2.1. С данными типа int производятся следующие действия: математические операции (инкремент в циклах с параметром), и хранение элементов (тип информ. поля – int).

for (inti = 2; i<=size; i++)

int data;

2.2. С данными типа List производятся следующие действия: создание двусвязного списка, удаление элементов, вывод на экран.

2.3. С данными типа list<int>производятся те же действия (создание двусвязного списка, удаление элементов, вывод на экран), что и с типом List, но они реализованы с использованием STL.

1. *Вид данных.*

3.1. Двусвязный список реализован в программе с помощью структуры и STL.

1. *Структура.*

4.1. В программе используется структура List, которая реализует двусвязный список. Поле data имеет тип int и предназначено для хранения данных. Поле next имеет тип List\* и предназначено для хранения указателя на следующий элемент списка. Поле prev имеет тип List\* и предназначено для хранения указателя на предыдущий элемент.

structList

{

List\* prev;

char data;

List\* next;

};

1. *Ввод и вывод.*

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

cout << "enter 1 st element:\t";

cin >> p->data;

5.2. Для ввода и вывода структур реализованы функции void fillListSTL(int size), void PrintlistSTL(), spisok\* make(int size), void print(spisok\* first). Функции с символами STL в названии предназначены для работы с двусвязным списком, реализованным через STL. Остальные предназначены для работы с двусвязным списком, реализованным через структуры.

void fillListSTL (int size)

{

int tmp;

cout << "Ваш список на " << size << " элементов:" << endl;

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

{

cout << "Введите " << i + 1 << " Элемент:\t";

cin >> tmp;

mylist.push\_back(tmp);

}

}

void PrintlistSTL()

{

list <int> ::iterator startpos = mylist.begin();

list <int> ::iterator endpos = mylist.end();

int i = 0;

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

{

cout << i+1 << " Элемент:\t" << \*startpos << endl;

i++;

}

}

spisok\* make(int size)

{

spisok\* first, \* p;

first = NULL;

p = new spisok;

cout << "Введите 1 элемент:\t";

cin >> p->data;

first = p;

p->next = NULL;

p->prev = NULL;

for (int i = 2; i <= size; i++)

{

spisok\* h = new spisok;

spisok\* q = p;

p->next = h;

p = p->next;

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

cin >> p->data;

p->next = NULL;

p->prev = q;

}

return first;

}

void print(spisok\* first)

{

if (first == NULL)

{

cout << "Список пуст " << endl;

}

else

{

spisok\* p = first;

while (p != NULL)

{

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

p = p->next;

}

cout << endl;

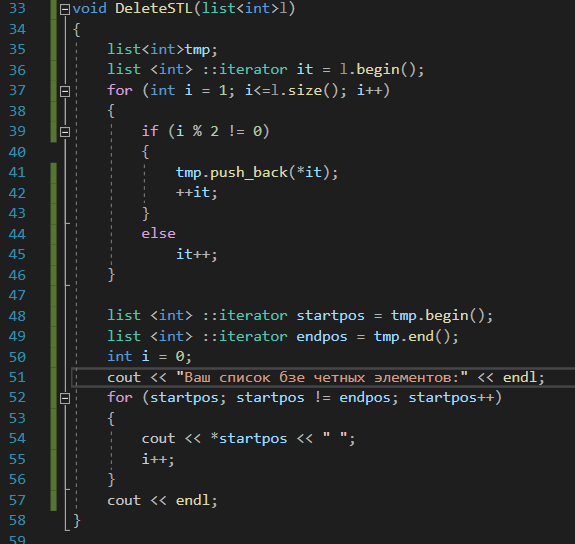
}

}

1. *Действия для решения задачи.*

6.1. Удаление элементов из списка через STL.

Передвижение по списку происходит через итератор, который увеличивается на каждой итерации цикла. Если элемент четный (по счету), то с ним ничего не происходит, а если нечетный, то он записывается в отдельный список (tmp).



6.2. Удаление элементов из списка, реализованного через структуру.

Удаление выполняется с помощью цикла , он проверяет является ли номер элемента четным, если да, то проверяется является ли он последним, если да, то происходит его удаление, заменяя значения указателей у предыдущего элемента, если нет, то происходит его удаление, заменяя значения указателей у соседних элементов. После чего перезаписываются соседние элементы для следующего добавления.

void DeleteElements(spisok\*& first, int size)

{

spisok\* tmp = first;

int i = 1;

while (tmp != nullptr)

{

if (i % 2 == 0)

{

if (tmp->next == nullptr)

{

spisok\* tmp2 = tmp->prev;

delete tmp;

tmp2->next = 0;

tmp = tmp2;

}

else

{

spisok\* tmp2 = tmp->prev;

spisok\* tmp3 = tmp->next;

delete tmp;

tmp2->next = tmp3;

tmp3->prev = tmp2;

tmp = tmp3;

}

}

else

{

tmp = tmp->next;

}

i++;

}

}

**Кодпрограммы**

#include <iostream>

#include<list>

#include<iterator>

using namespace std;

list<int>mylist;

void fillListSTL (int size)

{

int tmp;

cout << "Ваш список на " << size << " элементов:" << endl;

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

{

cout << "Введите " << i + 1 << " Элемент:\t";

cin >> tmp;

mylist.push\_back(tmp);

}

}

void PrintlistSTL()

{

list <int> ::iterator startpos = mylist.begin();

list <int> ::iterator endpos = mylist.end();

int i = 0;

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

{

cout << i+1 << " Элемент:\t" << \*startpos << endl;

i++;

}

}

list<int> DeleteSTL(list<int>l)

{

list <int> ::iterator startpos = l.begin();

list <int> ::iterator endpos = l.end();

int i = 1;

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

{

if (i % 2 == 0) {

l.erase(startpos);

l.size() - 1;

}

i++;

}

return l;

}

struct spisok

{

int data;

spisok\* next;

spisok\* prev;

};

spisok\* make(int size)

{

spisok\* first, \* p;

first = NULL;

p = new spisok;

cout << "Введите 1 элемент:\t";

cin >> p->data;

first = p;

p->next = NULL;

p->prev = NULL;

for (int i = 2; i <= size; i++)

{

spisok\* h = new spisok;

spisok\* q = p;

p->next = h;

p = p->next;

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

cin >> p->data;

p->next = NULL;

p->prev = q;

}

return first;

}

void print(spisok\* first)

{

if (first == NULL)

{

cout << "Список пуст " << endl;

}

else

{

spisok\* p = first;

while (p != NULL)

{

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

p = p->next;

}

cout << endl;

}

}

void DeleteElements(spisok\*& first, int size)

{

spisok\* tmp = first;

int i = 1;

while (tmp != nullptr)

{

if (i % 2 == 0)

{

if (tmp->next == nullptr)

{

spisok\* tmp2 = tmp->prev;

delete tmp;

tmp2->next = 0;

tmp = tmp2;

}

else

{

spisok\* tmp2 = tmp->prev;

spisok\* tmp3 = tmp->next;

delete tmp;

tmp2->next = tmp3;

tmp3->prev = tmp2;

tmp = tmp3;

}

}

else

{

tmp = tmp->next;

}

i++;

}

}

int main()

{

setlocale(LC\_ALL, "rus");

int size;

cin >> size;

fillListSTL(size);

PrintlistSTL();

DeleteSTL(mylist);

PrintlistSTL();

spisok\* list = make(size);

cout << "Ваш список:" << endl;

print(list);

DeleteElements(list, size);

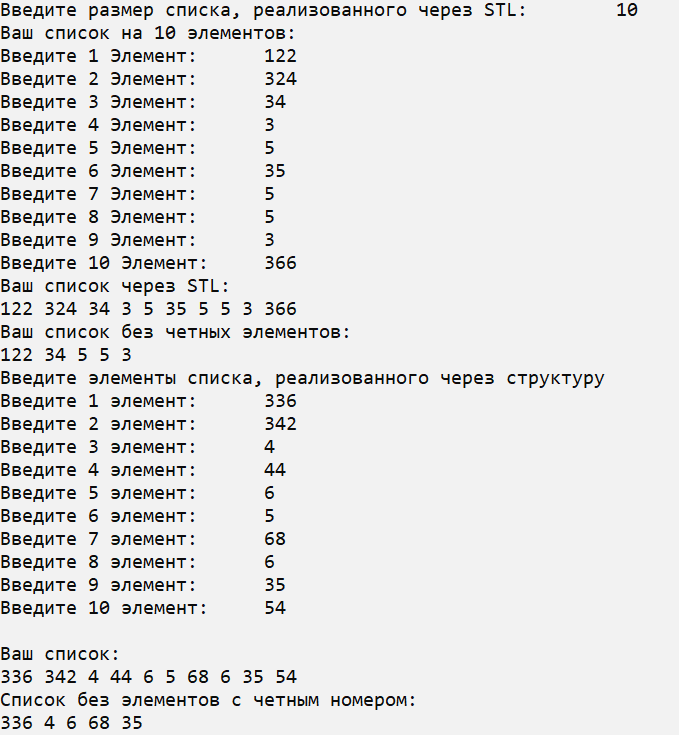
cout << "Список без элементов с четным номером:" << endl;

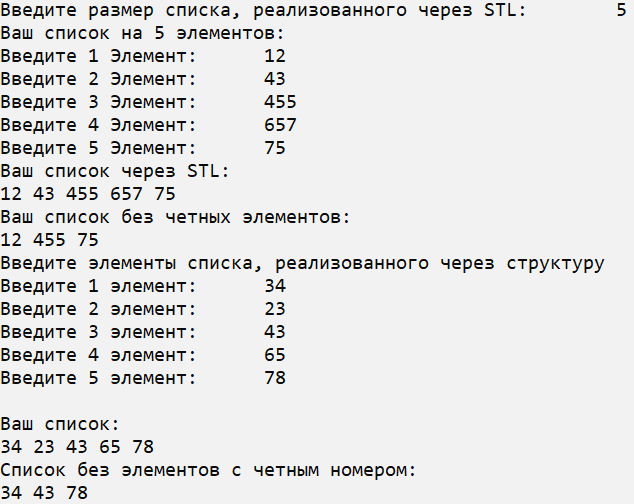
print(list);

system("pause");

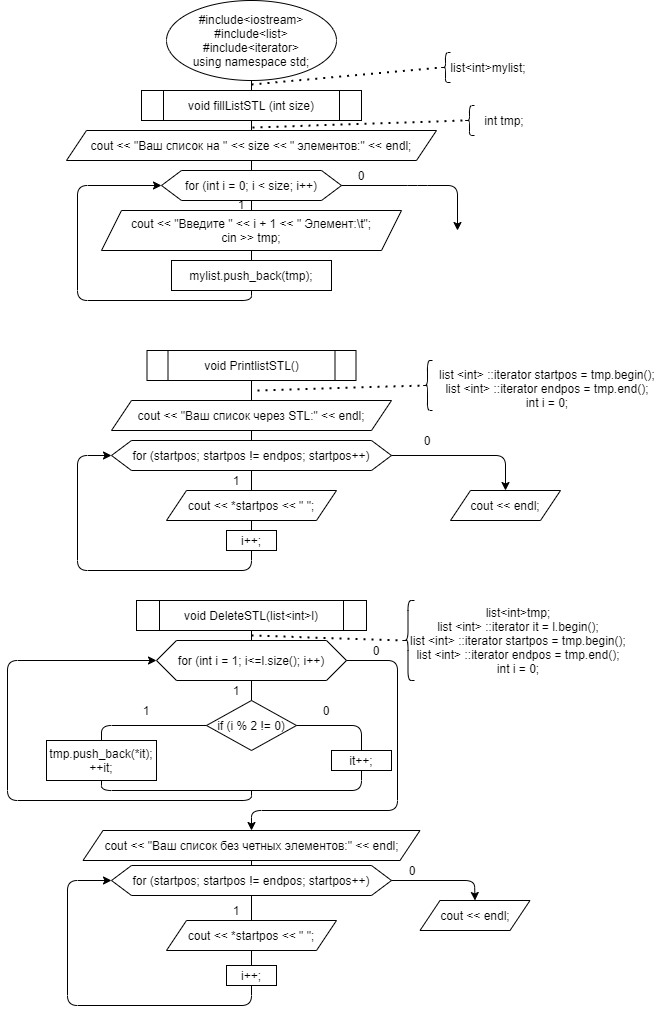
}

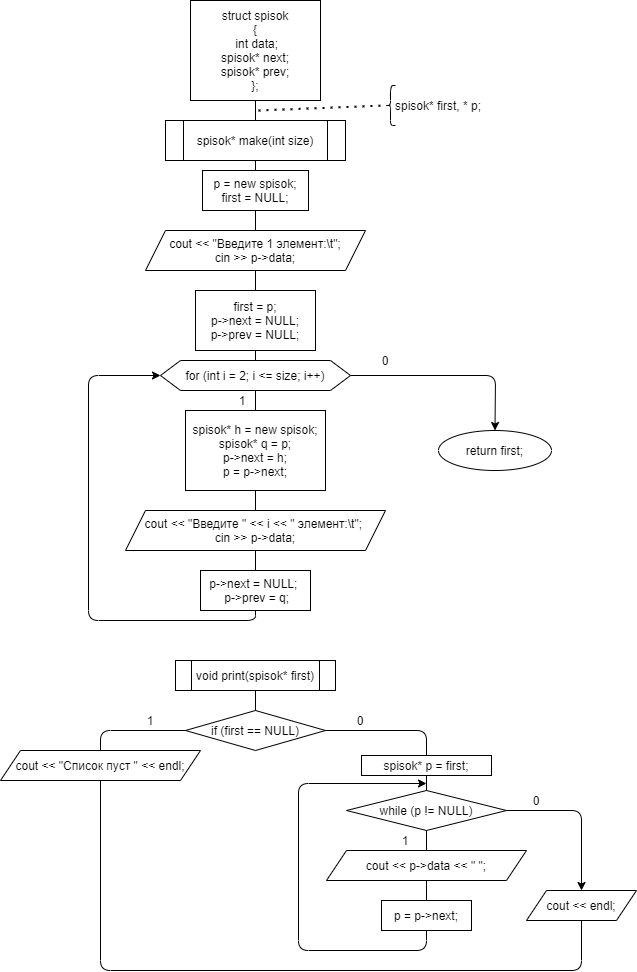
**Скриншоты**

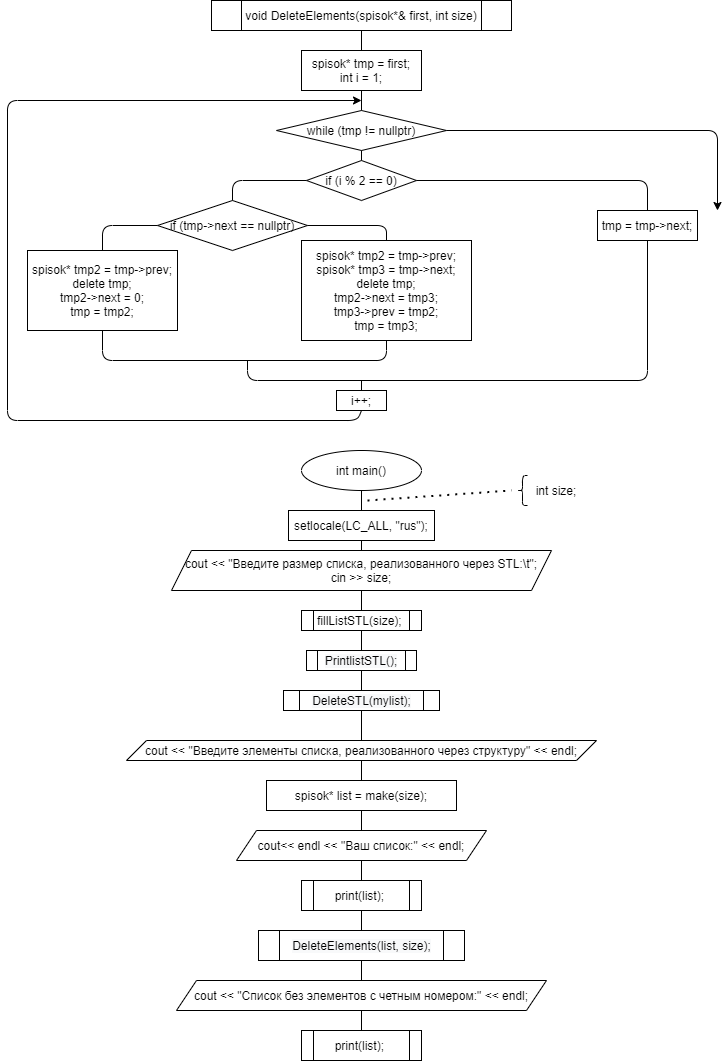
****

****

**Блок-схема**

****

****

****