Пермский Национальный Исследовательский Политехнический Университет

Лабораторная работа №11.1

"Динамические структуры данных"

21 вариант

Выполнил:

Студент группы ИВТ-20-2б

Галинов Олег Юрьевич

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 char\*. Добавить в список элемент после элемента с заданным информационным полем.)

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, char.

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 = 2; i <= size; i++)

С данными типа List производятся следующие действия: создание односвязного списка, добавление 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 char и предназначено для хранения значения. Поле next имеет тип List\* и предназначено для хранения указателя на следующий элемент списка.

struct List

{

char data;

List\* next;

};

1. *Ввод и вывод.*

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

cout << "Enter 1-st element:\t";

cin >> p->data;

1. *Действия для решения задачи.*
   1. Для создания списка используется цикл с параметром и оператор ветвления. Функция возвращает указатель на первый элемент.

List\* make(int size)

{

if (size == 0)

{

cout << "Список не может существовать!" << endl;

return 0;

}

List\* first, \* p;

first = NULL;

cout << "Введите 1 элемент:\t";

p = new List;

cin >> p->data;

first = p;

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

{

List\* h = new List;

p->next = h;

p = p->next;

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

cin >> p->data;

p->next = NULL;

}

return first;

}

* 1. Для печати списка используется оператор ветвления и цикл с предусловием.

void print(List\* first)

{

if (first == NULL)

{

cout << "Список пуст" << endl;

}

else

{

List\* p = first;

while (p != 0)

{

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

p = p->next;

}

cout << endl;

}

}

* 1. В добавлении элемента списка используются циклы с предусловием и цикл for(), чтобы дойти до того элемента, на место которого добавится новый.

void add(List\*& first)

{

if (first == NULL) return;

List\* p = first;

int size = 0;

while (p != NULL)

{

p = p->next;

size++;

}

p = first;

int pos;

cout << "Введите номер элемента, после которого хотите добавить элемент:\t";

cin >> pos;

while (pos > size || pos < 1)

{

cout << "Введите корректное число: ";

cin >> pos;

}

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

{

p = p->next;

}

List\* h = new List;

h->next = p->next;

p->next = h;

p = p->next;

cout << "Введите новый элемент:\t";

cin >> p->data;

}

* 1. Для ввода корректного размера списка используется функция getSize(), которая проверяет размерность списка (не меньше 1)

int getSize()

{

int num;

cin >> num;

while (num<1)

{

cout << "Введите корректное число: ";

cin >> num;

}

return num;

}

**Код программы**

#include <iostream>

using namespace std;

struct List

{

char data;

List\* next;

};

int getSize()

{

int num;

cin >> num;

while (num<1)

{

cout << "Введите корректное число: ";

cin >> num;

}

return num;

}

List\* make(int size)

{

if (size == 0)

{

cout << "Список не может существовать!" << endl;

return 0;

}

List\* first, \* p;

first = NULL;

cout << "Введите 1 элемент:\t";

p = new List;

cin >> p->data;

first = p;

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

{

List\* h = new List;

p->next = h;

p = p->next;

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

cin >> p->data;

p->next = NULL;

}

return first;

}

void print(List\* first)

{

if (first == NULL)

{

cout << "Список пуст" << endl;

}

else

{

List\* p = first;

while (p != 0)

{

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

p = p->next;

}

cout << endl;

}

}

void add(List\*& first)

{

if (first == NULL) return;

List\* p = first;

int size = 0;

while (p != NULL)

{

p = p->next;

size++;

}

p = first;

int pos;

cout << "Введите номер элемента, после которого хотите добавить элемент:\t";

cin >> pos;

while (pos > size || pos < 1)

{

cout << "Введите корректное число: ";

cin >> pos;

}

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

{

p = p->next;

}

List\* h = new List;

h->next = p->next;

p->next = h;

p = p->next;

cout << "Введите новый элемент:\t";

cin >> p->data;

}

int main()

{

system("chcp 1251 > null");

setlocale(LC\_ALL, "rus");

int size;

cout << "Введите размер списка\t";

size = getSize();

List\* list = make(size);

cout << "Ваш список:" << endl;

print(list);

add(list);

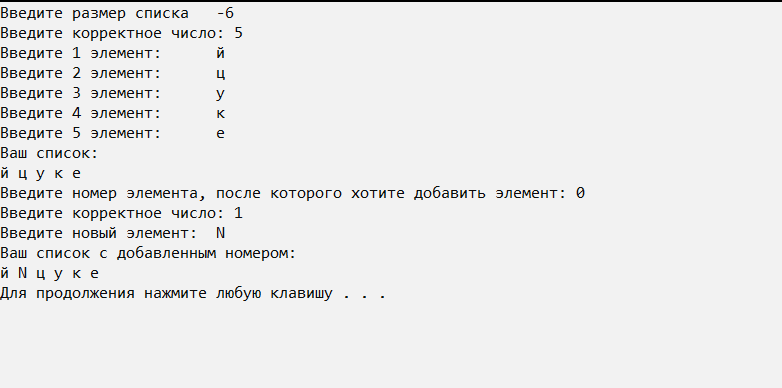
cout << "Ваш список с добавленным номером:" << endl;

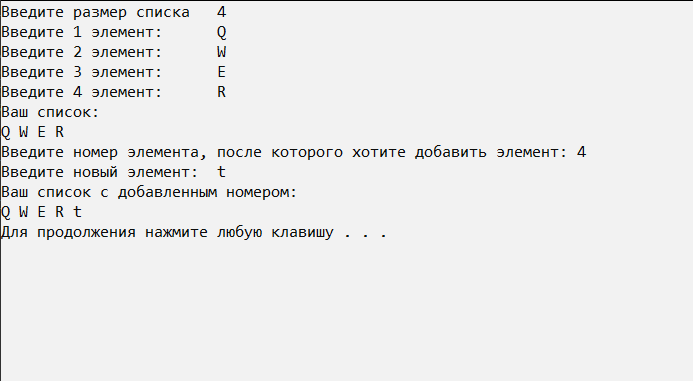
print(list);

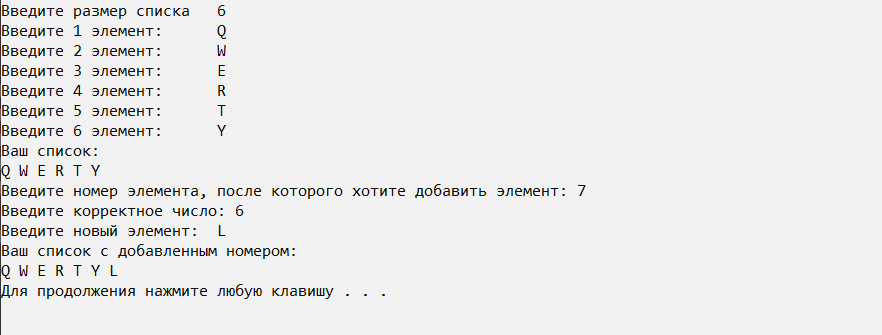
system("pause");

}

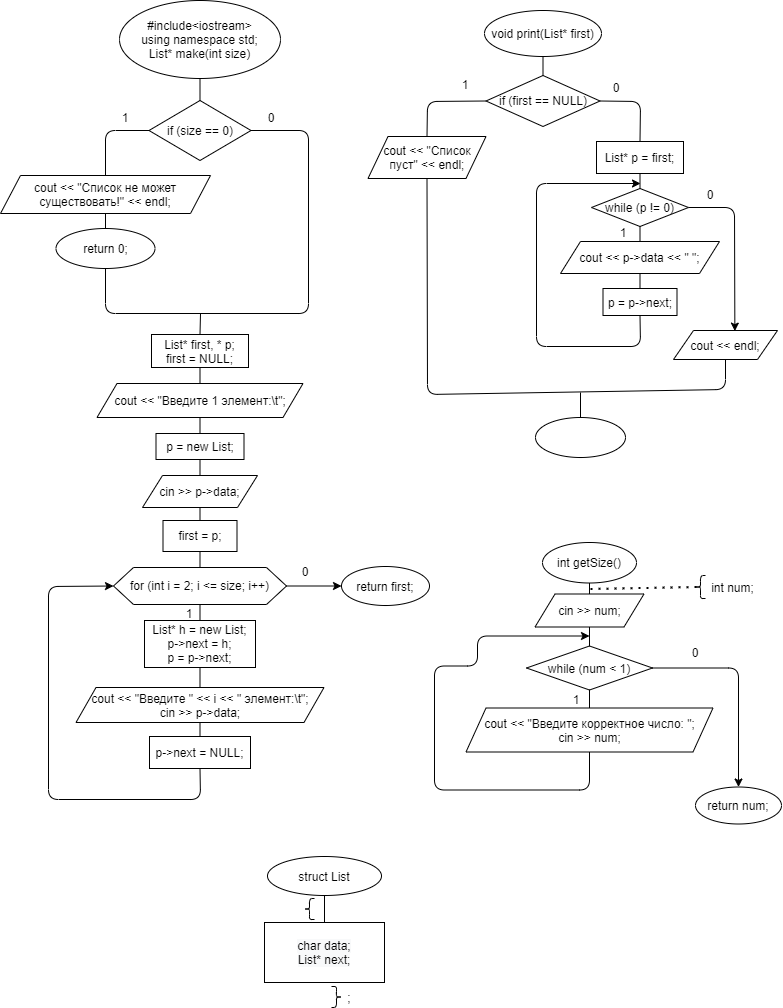
**Скриншоты**

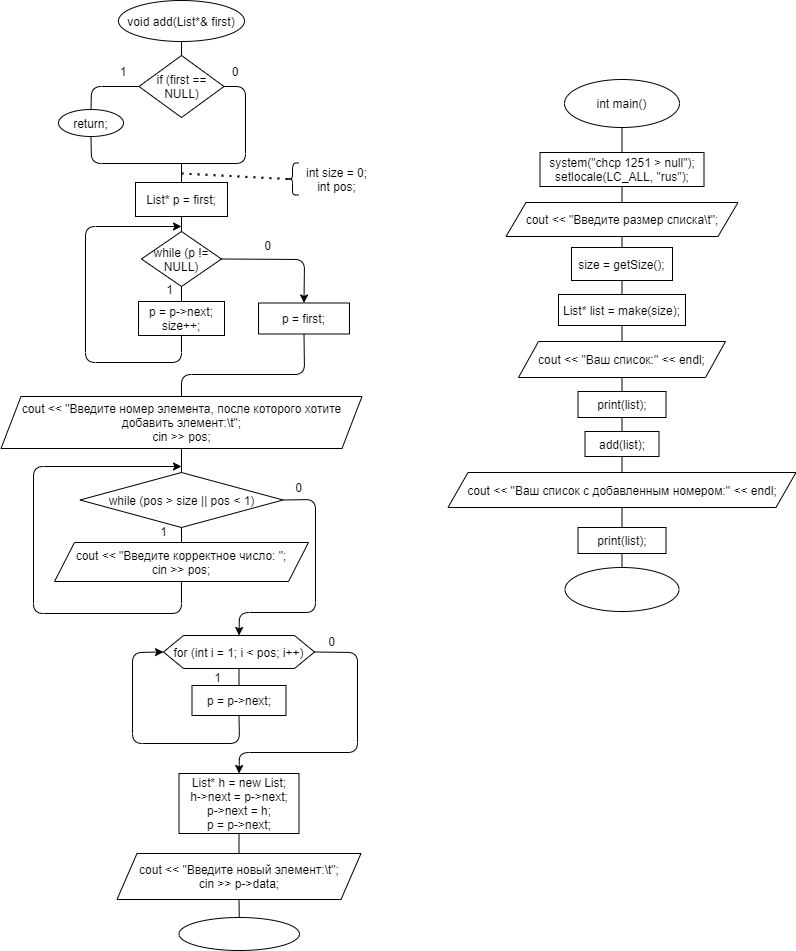
****

****

****

**Блок-схема**

****

****