**Отчет**

**по Лабораторной работе № 7**

**«Работа с динамическими структурами данных»**

Специальность 09.02.03«Программирование в компьютерных системах»

ПМ.01 «Разработка программных модулей и программного обеспечения для компьютерных систем»

МДК.01.02 «Прикладное программирование»

**Министерство образования и науки Российской Федерации**

**федеральное государственное автономное образовательное учреждение высшего образования**

**“САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ,**

**МЕХАНИКИ И ОПТИКИ”**

**ФАКУЛЬТЕТ СРЕДНЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

Выполнил:

студент группы 2337

Коряков С.А.

Санкт-Петербург

2018/2019

Преподаватель:

ФИО Сорокин Д. Ю.

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_201\_\_г.

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Цель работы

Продемонстрировать работа с динамическими структурами данных, на примере списков, очередей, стеков или бинарных деревьев.

Практическое задание

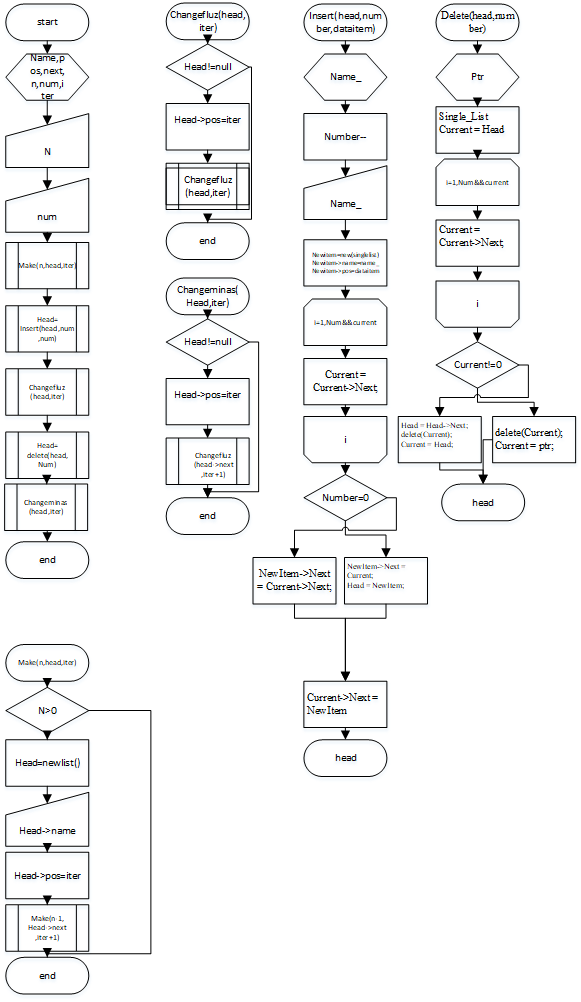
Реализовать программу согласно варианту задания .

Индивидуальное задание

8. Гаражная стоянка имеет одну стояночную полосу, причем единственный въезд и он же выезд находится в одном конце полосы. Если владелец автомобиля приходит забрать автомобиль, который не является ближайшим к выходу, то все автомобили, загораживающие проезд, удаляются, машина данного владельца выводится со стоянки, а другие машины возвращаются на стоянку в исходном порядке. Написать программу, которая моделирует процесс прибытия и отъезда машин. Прибытие или отъезд автомобиля задается командной строкой, которая содержит признак прибытия или отъезда и номер машины. Программа должна выводить сообщение при прибытии или выезде любой машины. При выезде автомобиля со стоянки сообщение должно содержать число случаев, когда машина удалялась со стоянки для обеспечения выезда других автомобилей.

Блок-схема алгоритма работы программы

Рисунок 1, алгоритм выполнения программы



Листинг

#include <iostream> // подключение основной библиотеки для работы основных функций

#include <string>

using namespace std; // использование постоянного пространства имен для удобного пользования

struct Single\_List {//структура данных

string name;//имя машины

int pos;//ее позиция

Single\_List \*Next; //адресное поле

};

Single\_List \*Head;//экземпляры ф-ций

void Make\_Single\_List(int n, Single\_List\*\* Head, int iter);

void Print\_Single\_List(Single\_List\* Head);

Single\_List\* Insert\_Item\_Single\_List(Single\_List\* Head, int Number, int DataItem);

void Delete\_Single\_List(Single\_List\* Head);

void changeFluz(Single\_List\* Head, int iter);

Single\_List\* Delete\_Item\_Single\_List(Single\_List\* Head, int Number);

void changeMinas(Single\_List\* Head, int iter);

void Viezd(Single\_List\* Head, int iter, int num);

int main()

{

int n, num, iter=1,t=1;

cout << "vvedite k ol-vo mashin an avtosotyanke: ";

cin >> n;

Make\_Single\_List(n, &Head, iter);//вызов функции по созданию однонаправленного списка

Print\_Single\_List(Head);//печать однонаправленного списка

while (t != 0)

{

cout << "Chto hotite sdelat? 1 - sovershit v'ezd, 2 - sovershit viezd, 0 - zavershit rabotu: ";

cin >> t;

if (t == 1)

{

cout << "Vvedite Nomer mesta dlya v'ezda: ";

cin >> num;

Head = Insert\_Item\_Single\_List(Head, num, num);//добавление машины в однонаправелнный список

Viezd(Head, iter, num);//отображение выехавших машин

cout << "\n";

changeFluz(Head, iter);//новый вид списка после добавленния

cout << "\n";

}

else if (t == 2)

{

cout << "Vvedite Nomer mesta dlya viezda: ";

cin >> num;

Head = Delete\_Item\_Single\_List(Head, num);//удаление машины из однонаправелнного списка

Viezd(Head, iter, num);//отображение выехавших машин

cout << "\n";

changeMinas(Head, iter);//новый вид списка после убавленя

cout << "\n";

}

else if (t != 0)

{

cout << "Vvedite corretniy nomer";

cout << "\n";

}

}

Delete\_Single\_List(Head);//очищение памяти от списка

}

void Make\_Single\_List(int n, Single\_List\*\* Head,int iter) {

if (n > 0) {

(\*Head) = new Single\_List();

//выделяем память под новый элемент

cout << "Vedite imya machini ";

cin >> (\*Head)-> name;

(\*Head)->pos = iter;

//вводим значение информационного поля

(\*Head)->Next = NULL;//обнуление адресного поля

Make\_Single\_List(n - 1, &((\*Head)->Next), iter+1);

}

}

void Print\_Single\_List(Single\_List\* Head) {

if (Head != NULL) {

cout<<"|Name: " << Head->name <<", Position: " << Head->pos << "|\t";

Print\_Single\_List(Head->Next);

//переход к следующему элементу

}

else cout << "\n";

}

void changeFluz (Single\_List\* Head, int iter) {

if (Head != NULL) {

Head->pos = iter;

cout << "|Name: " << Head->name << ", Position: " << Head->pos << "|\t";

changeFluz(Head->Next, iter + 1);

//переход к следующему элементу

}

}

void Viezd(Single\_List\* Head, int iter, int num) {

if ((Head != NULL)&(num!=iter)) {

Head->pos = iter;

cout << "|Name: " << Head->name << ", Position: " << Head->pos << " viehal|\t";

Viezd(Head->Next, iter + 1, num);

//переход к следующему элементу

}

}

void changeMinas(Single\_List\* Head, int iter)

{

if (Head != NULL) {

Head->pos = iter;

cout << "|Name: " << Head->name << ", Position: " << Head->pos << "|\t";

changeMinas(Head->Next, iter + 1 );

//переход к следующему элементу

}

}

void Delete\_Single\_List(Single\_List\* Head) {

if (Head != NULL) {

Delete\_Single\_List(Head->Next);

delete Head;

}

}

Single\_List\* Insert\_Item\_Single\_List(Single\_List\* Head,int Number, int DataItem)

{

Number--;

string name\_;

Single\_List \*NewItem = new(Single\_List);

cout << "Vedite imya machini ";

cin >> name\_;

NewItem->name = name\_;

NewItem->pos = DataItem;

NewItem->Next = NULL;

Single\_List \*Current = Head;

for (int i = 1; i < Number && Current->Next != NULL; i++)

{

Current = Current->Next;

}

if (Number == 0) {

//вставляем новый элемент на первое место

NewItem->Next = Current;

Head = NewItem;

;

}

else {//вставляем новый элемент на непервое место

if (Current->Next != NULL)

{

NewItem->Next = Current->Next;

}

Current->Next = NewItem;

}

return Head;

}

Single\_List\* Delete\_Item\_Single\_List(Single\_List\* Head, int Number) {

Single\_List \*ptr;//вспомогательный указатель

Single\_List \*Current = Head;

for (int i = 1; i < Number && Current != NULL; i++)

{

Current = Current->Next;

}

if (Current != NULL) {//проверка на корректность

if (Current == Head) {//удаляем первый элемент

Head = Head->Next;

delete(Current);

Current = Head;

}

else {//удаляем непервый элемент

ptr = Head;

while (ptr->Next != Current)

{

ptr = ptr->Next;

}

ptr->Next = Current->Next;

delete(Current);

Current = ptr;

}

}

return Head;}

Протокол выполнения



Рисунок 2, протокол выполнения программы