Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Системы автоматизированного проектирования»

**ОТЧЕТ**

по лабораторной работе №5

по курсу «Информационное обеспечение САПР»

на тему: «Разработка абстрактного типа данных и его реализацию

в вид параметризованного класса»

Выполнили студенты

группы 16ВВ3:

Страхов А.В.

Самушкин А.Д.

Борисов Д.А.

Приняли:

Глотова Т.В.

Евсеева Ю.И.

Пенза 2018

**Цель работы:** изучение методов организации хранения и обработки данных на примере списочных структур данных.

**Задание:**

Разработать АТД (абстрактный тип данных) и его реализацию в виде параметризованного класса.

Вариант 2. Двунаправленный список.

Реализуемые конструкции:

1. Добавление;
2. Удаление;
3. Вывод список с начала;
4. Вывод списка с конца;
5. Сохранение в файл;
6. Определение максимального элемента списка;
7. Определение минимального элемента списка.

**Ход работы:**

Алгоритм работы пользовательского интерфейса консольного приложения схематически представлен на рисунке 1.

Работа АТД обусловлена созданным классом List в основу которого ложится структура данных «struct», включающая в себя:

1. Имя переменной;
2. Указатель на следующий элемент;
3. Указатель на предыдущий элемент.

Работа со структурой описана в «Публичном» разделе класса «List», к каждому пункту из задания была создана своя функция по обработке списка.

Подробнее с описанием класса и самой программой можно ознакомиться, перейдя в раздел «Листинг».

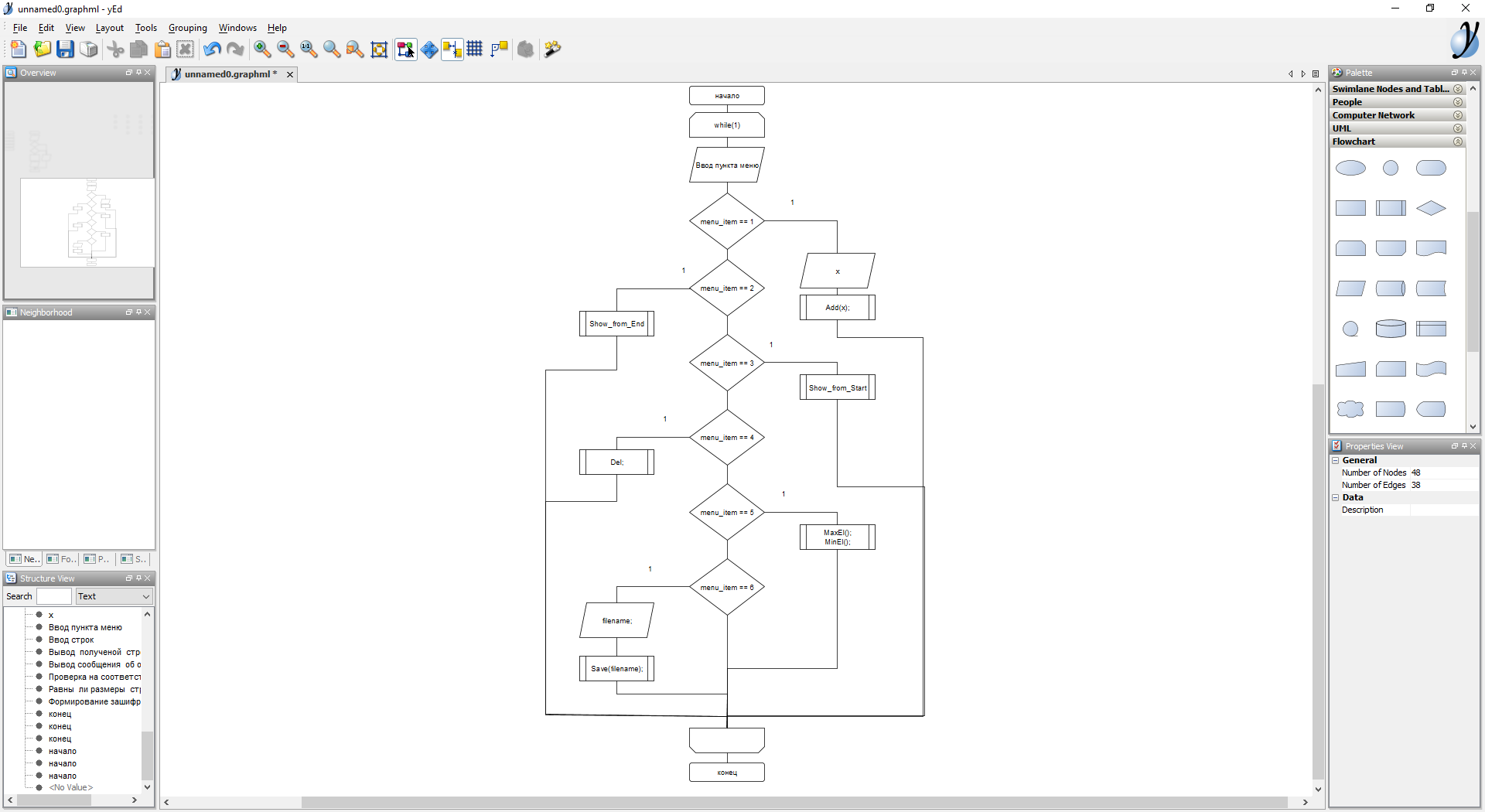


Рисунок 1

**Пример работы приложения:**



Рисунок 2 – Основное окно приложения



Рисунок 3 – Пример добавления переменной в список



Рисунок 4 – Пример вывода элементов по порядку



Рисунок 5 – Пример вывода списка в обратном порядке

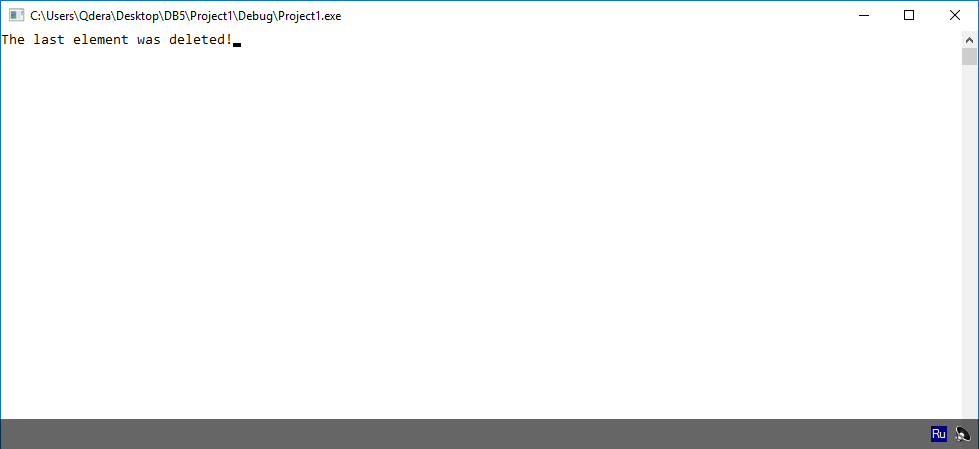


Рисунок 6 – Сообщение о удаление последнего элемента

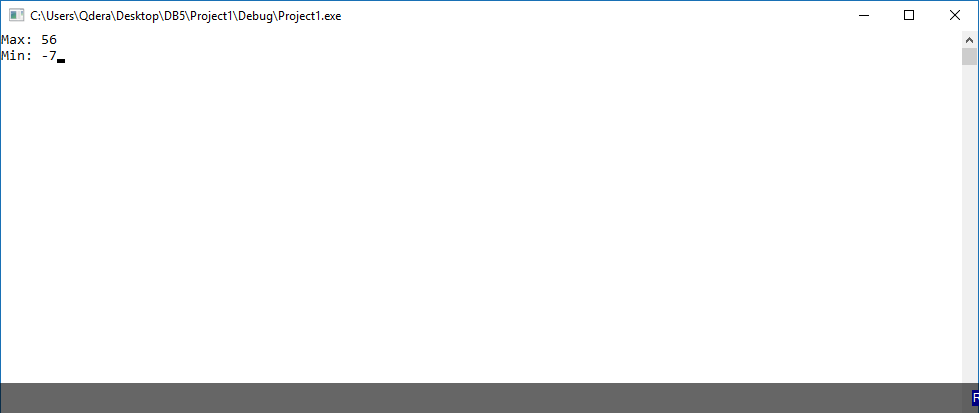


Рисунок 7– Поиск макс. и мин. элементов в списке

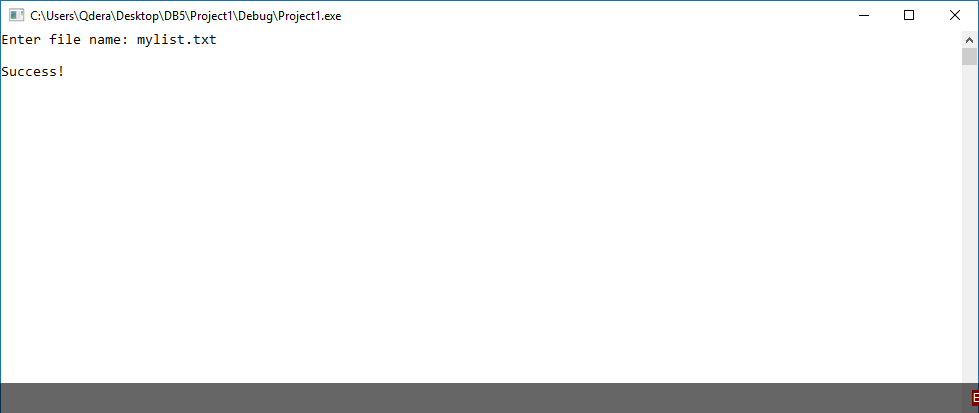


Рисунок 8 – Сообщение об успешном сохранении списка в файл

**Листинг:**

#include <iomanip>

#include <iostream>

#include <string>

#include <conio.h>

#include <fstream>

using namespace std;

class List //Создаем тип данных Список

{

struct Node //Структура, являющаяся звеном списка

{

int x; //Значение x будет передаваться в список

Node \*Next, \*Prev; //Указатели на адреса следующего и предыдущего элементов списка

};

Node \*Head, \*Tail; //Указатели на адреса начала списка и его конца

public:

List() :Head(NULL), Tail(NULL) {}; //Инициализируем адреса как пустые

~List(); //Прототип деструктора

void Add(int x);

void Show\_from\_start();

void Show\_from\_end();

void Del();

int MaxEl();

int MinEl();

void Save(string filename);

};

int main()

{

List lst;

while (1) {

system("CLS");

cout << "Choose menu item:\n"

<< "1.Add;\n"

<< "2.Show from end;\n"

<< "3.Show from start;\n"

<< "4.Delete;\n"

<< "5.Max&Min;\n"

<< "6.Save;\n"

<< "->";

string menu\_item;

cin >> menu\_item;

system("CLS");

if (menu\_item == "1") {

cout << "Write the number -> ";

int what\_to\_add;

cin >> what\_to\_add;

cout << endl;

lst.Add(what\_to\_add);

cout << "The number was added!";

}

else if (menu\_item == "2") {

cout << "Current list:\n";

lst.Show\_from\_end();

}

else if (menu\_item == "3") {

cout << "Current list:\n";

lst.Show\_from\_start();

}

else if (menu\_item == "4") {

lst.Del();

cout << "The last element was deleted!";

}

else if (menu\_item == "5") {

cout << "Max: " << lst.MaxEl();

cout << "\nMin: " << lst.MinEl();

}

else if (menu\_item == "6") {

string fname;

cout << "Enter file name: ";

cin >> fname;

lst.Save(fname);

cout << "\nSuccess!";

}

\_getch();

}

system("PAUSE");

return 0;

}

List::~List() //Деструктор

{

while (Head){

Tail = Head->Next; //Резервная копия адреса следующего звена списка

delete Head; //Очистка памяти от первого звена

Head = Tail; //Смена адреса начала на адрес следующего элемента

}

}

void List::Add(int x)

{

Node \*temp = new Node; //Выделение памяти под новый элемент структуры

temp->Next = NULL; //Указываем, что изначально по следующему адресу пусто

temp->x = x; //Записываем значение в структуру

if (Head != NULL){

temp->Prev = Tail; //Указываем адрес на предыдущий элемент в соотв. поле

Tail->Next = temp; //Указываем адрес следующего за хвостом элемента

Tail = temp; //Меняем адрес хвоста

} else {

temp->Prev = NULL; //Предыдущий элемент указывает в пустоту

Head = Tail = temp; //Голова=Хвост=тот элемент, что сейчас добавили

}

}

void List::Show\_from\_start()

{

Node \*temp = Head;

while (temp != NULL){

cout << temp->x << ' ';

temp = temp->Next;

}

}

void List::Show\_from\_end()

{

Node \*temp = Tail; //Временный указатель на адрес последнего элемента

while (temp != NULL){

cout << temp->x << ' '; //Выводить значение на экран

temp = temp->Prev; //Указываем, что нужен адрес предыдущего элемента

}

}

void List::Del()

{

Node \*temp;

Node \*current = Tail;

current = Head;

temp = Head->Next;

delete current;

Head = temp;

temp->Prev = NULL;

}

void List::Save(string filename)

{

ofstream save;

save.open(filename);

Node \*temp = Tail;

while (temp != NULL){

save << temp->x << endl;

temp = temp->Prev;

}

save.close(); save.clear();

}

int List::MaxEl()

{

Node \*temp = Tail;

int max = 0;

while (temp != NULL){

if (temp->x > max)

max = temp->x;

temp = temp->Prev;

}

return max;

}

int List::MinEl()

{

Node \*temp = Tail;

int min = MaxEl();

while (temp != NULL){

if (temp->x < min)

min = temp->x;

temp = temp->Prev;

}

return min;

}

**Вывод:** в этой работе мы изучили методы организации хранения и обработки данных на примере списочных структур данных.