Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе № 3

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Динамические списки»

Выполнили:

студенты группы 22ВВВ2:

Зубриянова А.А.

Кондратьева В.И.

Приняли:

Акифьев И.В.

Митрохин М.А.

Пенза 2023

**Название**

Динамические списки.

**Цель работы**

Изучить работу динамических списков.

**Лабораторное задание**

1. Реализовать приоритетную очередь, путём добавления элемента в список в соответствии с приоритетом объекта (т.е. объект с большим приоритетом становится перед объектом с меньшим приоритетом).
2. \* На основе приведенного кода реализуйте структуру данных Очередь.
3. \* На основе приведенного кода реализуйте структуру данных Стек.

**Описание метода решения задачи**

Во всех заданиях используются функции:

add\_elem, с помощью которой пользователь вводит имя и приоритет структуры.

review, которая выводит приоритетную очередь/очередь/стек (просмотр содержимого списка).

free\_list, которая освобождает приоритетную очередь /очередь/стек.

Задание1:

Приоритетная очередь.

Запись (добавление элемента) в любое место списка по приоритету - функция add\_to\_list\_pr.

Извлечение (удаление элемента) из конца списка - функция del\_from\_list. При извлечении удаляется элемент с самым низким приоритетом.

Задание 2:

Очередь.

Запись (добавление элемента) в начало списка - функция add\_to\_list\_queue.

Извлечение (удаление элемента) из конца списка - функция del\_from\_list. При извлечении удаляется элемент, который добавили самым первым.

Задание 3:

Стек.

Запись (добавление элемента) в конец списка - функция add\_to\_list.

Извлечение (удаление элемента) из конца списка - функция del\_from\_list. При извлечении удаляется элемент, который добавили самым последним.

**Листинг**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <windows.h>

void add\_to\_list(void);

void add\_to\_list\_pr(void);

void add\_to\_list\_queue(void);

void del\_from\_list(void);

struct node\* add\_elem(void);

void review(void);

void free\_list(void);

struct node\* head = NULL, \* last = NULL;

struct node

{

char inf[100]; // полезная информация

int prior;

struct node\* next; // ссылка на следующую такую же структуру

};

int main()

{

setlocale(LC\_ALL, "Russian");

int N;

printf("Введите размер списка: ");

scanf("%d", &N);

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

{

add\_to\_list\_pr(); // очередь с приоритетом (1)

add\_to\_list\_queue(); // очередь (2)

add\_to\_list(); // стек (3)

}

review();

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

{

printf("Удаление последнего элемента\n");

//if (i % 3 == 0)

//{

// add\_to\_list\_pr(); // очередь с приоритетом (1)

// add\_to\_list\_queue(); // очередь (2)

// add\_to\_list(); // стек (3)

//}

del\_from\_list();

review();

}

free\_list();

return 0;

}

void add\_to\_list(void)

{

struct node\* p = NULL;

p = add\_elem();

if (head == NULL) // если списка нет, то устанавливаем голову списка

{

head = p;

last = p;

}

else // если список уже есть, то вставляем в конец

{

last->next = p;

last = p;

}

}

void add\_to\_list\_pr(void)

{

struct node\* p = NULL;

p = add\_elem();

if (head == NULL) // если списка нет, то устанавливаем голову списка

{

head = p;

last = p;

}

else // если список уже есть, то вставляем по приоритету

{

if (head->prior < p->prior) // вставляем в начало (т.е объект с большим приоритетом перед объектом с меньшим приоритетом)

{

p->next = head;

head = p;

}

else

{

struct node\* struc = head;

while (struc)

{

if (struc->next == NULL) // вставляем в конец

{

struc->next = p;

p->next = NULL;

last = p;

break;

}

else if (struc->next->prior < p->prior) // вставляем в середину

{

p->next = struc->next;

struc->next = p;

break;

}

struc = struc->next;

}

}

}

}

void add\_to\_list\_queue(void)

{

struct node\* p = NULL;

p = add\_elem();

if (head == NULL) // если списка нет, то устанавливаем голову списка

{

head = p;

last = p;

}

else // если список уже есть, то вставляем в начало

{

p->next = head;

head = p;

}

}

void del\_from\_list(void)

{

struct node\* struc = head;

if (struc->next == NULL) // удаляем единственный элемент

{

free(struc);

head = NULL;

last = NULL;

return;

}

while (struc)

{

if (struc->next == last) // удаляем последний элемент

{

free(struc->next);

struc->next = NULL;

last = struc;

break;

}

struc = struc->next;

}

}

struct node\* add\_elem(void) // Создаем новый элемент

{

struct node\* p = NULL;

char s[100];

int pr;

p = (node\*)malloc(sizeof(struct node)); // выделяем память под новый элемент списка

if (p == NULL) // если память не выделилась

{

printf("Ошибка при распределении памяти\n");

exit(1);

}

printf("Введите название объекта: \n"); // вводим название

scanf("%s", s);

strcpy(p->inf, s);

printf("Введите приоритет объекта: \n"); // вводим приоритет

scanf("%d", &pr);

p->prior = pr;

p->next = NULL;

return p; // возвращаем указатель на созданный элемент (где его создали, в каком месте памяти)

}

void review(void)

{

struct node\* struc = head;

if (head == NULL)

{

printf("Список пуст\n");

}

while (struc)

{

printf("Имя - %s, " "Приоритет - %d", struc->inf, struc->prior);

struc = struc->next;

printf("\n");

}

}

void free\_list(void)

{

struct node\* struc = head, \* p;

while (struc)

{

p = struc->next; // p указывает на следующий

free(struc); // освобождаем текущий

struc = p; // struc переходит на p, т.е на следующий

}

}

**Результаты работы программы**

Задание 1:



Рисунок 1 – Приоритетная очередь

Задание 2:

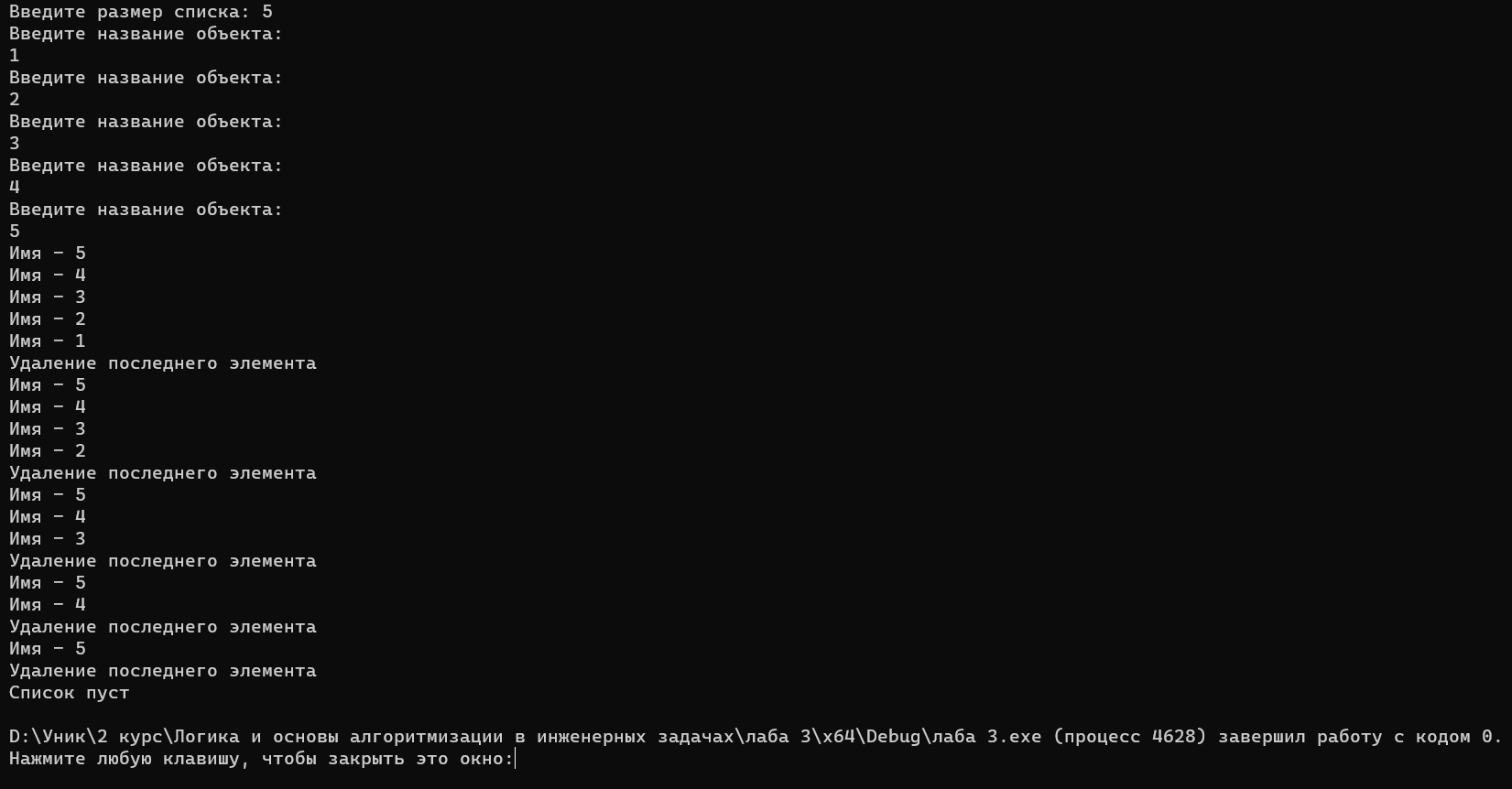


Рисунок 2 – Очередь

Задание 3:



Рисунок 3 – Стек

**Вывод**

В ходе выполнения лабораторной работы была разработана программа с использованием динамических структур данных. Результаты работы программы совпали с ожидаемыми результатами, следовательно, программа работает без ошибок.

Получили опыт в создании проектов в среде Microsoft Visual Studio, приобрели навыки программирования алгоритмов.