Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №3

по курсу “Логика и основа алгоритмизации в ИЗ”

на тему “ Динамические списки”

Выполнили студенты группы 21ВВ3:

Тюкалов В.Е.

Чинов Д.Д.

Приняли:

Митрохин М.А., Юрова О.В.

Пенза 2022

**Название**

Динамические списки.

**Цель работы:** выполнить лабораторные указания 1-3 реализуя приоритетную очередь, путем добавления элемента в список в соответствии с приоритетом объекта.

**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*.

**Задание 1:**

**Листинг**

// Выполнили студенты группы 21ВВ3: Тюкалов В.Е. и Чинов Д.Д.

// Приоритетная очередь

#include<iostream>

struct node

{

int pri;

char inf[256]; // полезная информация

struct node\* next = NULL; // ссылка на следующий элемент

};

struct node\* head = NULL, \* last = NULL, \* f = NULL; // указатели на первый и последний элементы списка

int dlinna = 0;

char find\_el[256];

int prioritet = 0;

struct node\* priority(node\* el);

struct node\* get\_struct(void)

{

struct node\* p = NULL;

char s[256];

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

{

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

exit(1);

}

puts("Введите название объекта: "); // вводим данные

std::cin >> s;

puts("Введите приоритет: ");

std::cin >> p->pri;

if (\*s == 0)

{

printf("Запись не была произведена\n");

return NULL;

}

strcpy\_s(p->inf, s);

p->next = NULL;

return p; // возвращаем указатель на созданный элемент

}

/\* Последовательное добавление в список элемента (в конец)\*/

void push(void)

{

struct node\* p = NULL;

p = get\_struct();

priority(p);

}

void pop(void)

{

struct node\* sw = NULL;

sw = head;

head = head->next;

free(sw);

}

/\* Просмотр содержимого списка. \*/

void review(void)

{

struct node\* struc = head;

if (head == NULL)

{

printf("Список пуст\n");

}

while (struc)

{

printf("Имя - %s, \n", struc->inf);

struc = struc->next;

}

printf("\n");

}

struct node\* priority(node\* el)

{

struct node\* sw = NULL;

if (head == NULL)

{

head = el;

last = el;

return NULL;

}

if (head->pri > el->pri)

{

el->next = head;

head = el;

return NULL;

}

struct node\* struc = head;

while (struc)

{

if (el->pri < struc->pri)

{

el->next = struc;

sw->next = el;

break;

}

sw = struc;

struc = struc->next;

}

return NULL;

}

int main()

{

setlocale(LC\_ALL, "RUS");

int var = 0;

printf("Кол-во данных: ");

std::cin >> var;

for (int i = 0; i < var; i++) {

push();

}

review();

pop();

review();

return 0;

}

**Результаты работы программы:**

Результаты работы программы показан на рисунке 1.

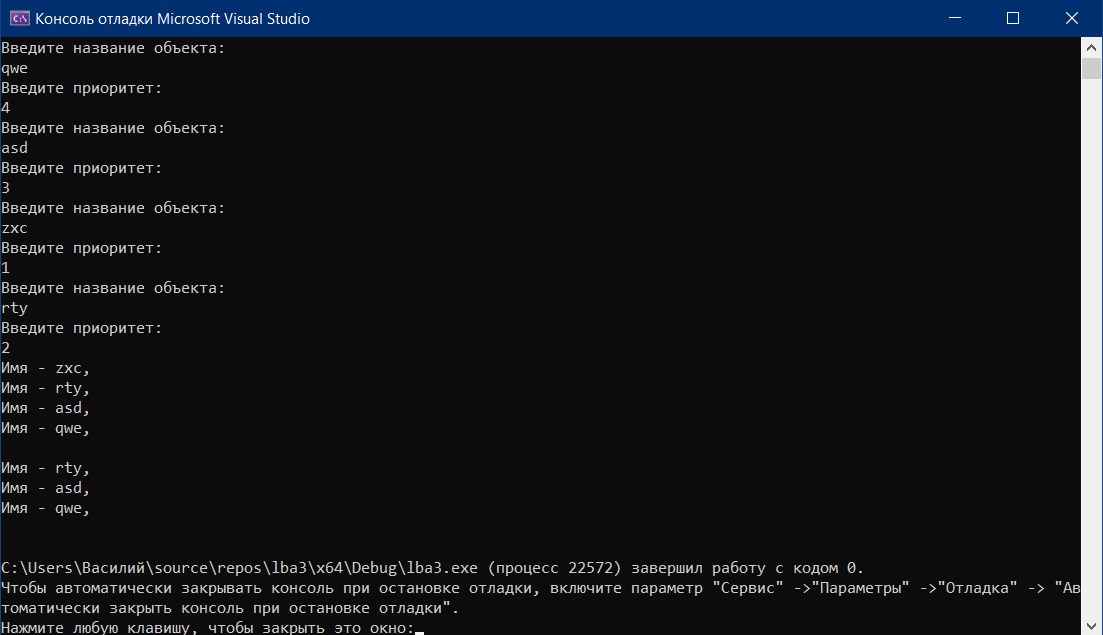


Рис. 1

Протокол трассировки программы показан на рисунке 2.

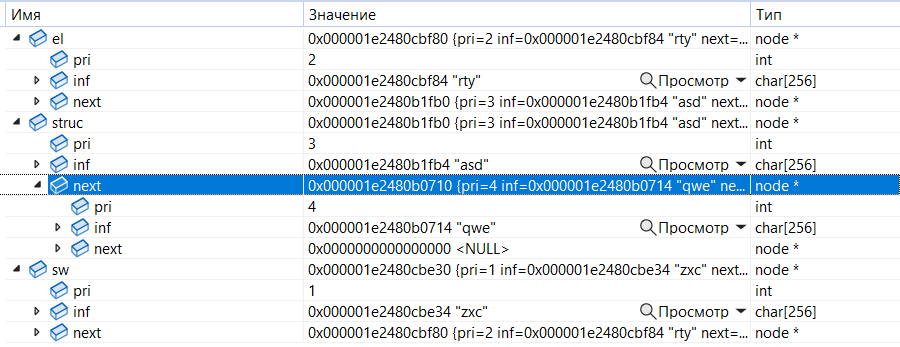


Рис. 2

Результат получился правильный, все действия совершены верно.

Результат работы программы, показанный на рисунке 1, совпал с результатами трассировки на рисунке 2.

**Задание 2 и 3:**

**Листинг**

//Алоритмы: Очередь и Стек

#include<iostream>

#include<string>

struct node

{

char inf[256]; // полезная информация

struct node\* next; // ссылка на следующий элемент

struct node\* prev; // ссылка на предыдущий элемент

};

struct node\* head = NULL, \* last = NULL, \* f = NULL; // указатели на первый и последний элементы списка

int dlinna = 0;

struct node\* get\_struct(void)

{

struct node\* p = NULL;

char s[256];

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

{

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

exit(1);

}

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

std::cin >> s;

if (\*s == 0)

{

printf("Запись не была произведена\n");

return NULL;

}

strcpy\_s(p->inf, s);

p->next = NULL;

p->prev = NULL;

return p; // возвращаем указатель на созданный элемент

}

void push(void)

{

struct node\* p = NULL;

p = get\_struct();

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

{

head = p;

last = p;

}

else if (head != NULL && p != NULL) // список уже есть, то вставляем в конец

{

last->next = p;

p->prev = last;

last = p;

}

return;

}

void pop\_stack(void)

{

struct node\* sw = NULL;

sw = last;

last = last->prev;

last->next = NULL;

free(sw);

}

void pop\_queue(void)

{

struct node\* sw = NULL;

sw = head;

head = head->next;

head->prev = NULL;

free(sw);

}

/\* Просмотр содержимого списка. \*/

void review(void)

{

struct node\* struc = head;

if (head == NULL)

{

printf("Список пуст\n");

}

while (struc)

{

printf("Имя - %s, \n", struc->inf);

struc = struc->next;

}

printf("\n");

}

int main(void)

{

setlocale(LC\_ALL, "rus");

push();

push();

push();

review();

pop\_queue();

review();

push();

review();

pop\_stack();

review();

}

**Результаты работы программы:**

Результаты работы программы показан на рисунке 1.

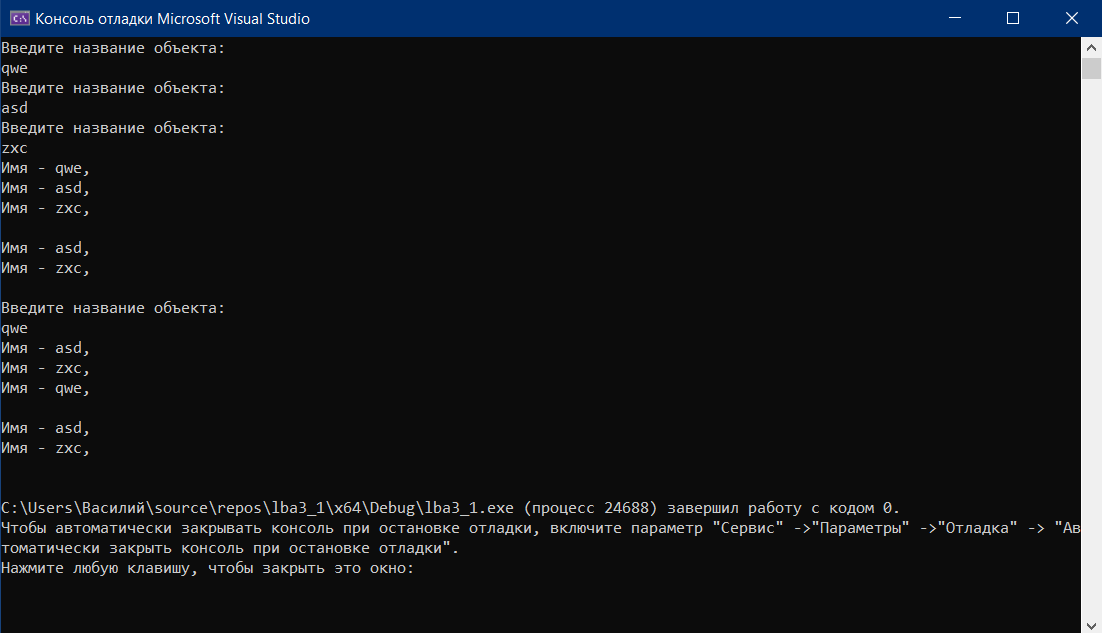


Рис. 1

Протокол трассировки программы показан на рисунке 2 и 3.

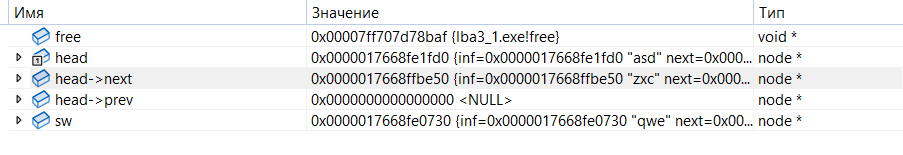


Рис. 2

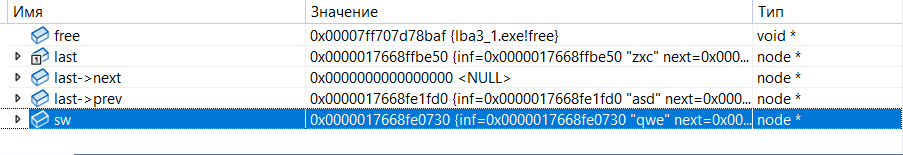


Рис. 3

Результат получился правильный, все действия совершены верно.

Результат работы программы, показанный на рисунке 1, совпал с результатами трассировки на рисунке 2 и 3

**Вывод:** мы выполнили лабораторные указания и на практике познакомились с такими структурами данных, как стек, очередь и приоритетная очередь.