Министерство науки и высшего 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 21вв1:**

Нечаев А.Д.

Киреев Д.А.

**Приняли**

Юрова О.В.

Акифьев И.В.

Пенза 2022

**Методические указания.**

**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:**

**Задание 1-2:**

Представили односвязный линейный список в виде структуры typedef struct node. Дальше с помощью функции struct node \*add(void) добавляем элементы в список. С помощью функции void spstore мы можем добавить элемент в любое место. Также организовали приоритетную очередь с приоритетом и структуру данных очередь. Функция void rewiew(void) выводит список в консоль.

**Задание 3:**

Представили двухсвязный линейный список в виде структуры typedef struct node. Дальше с помощью функции struct node \*add(void) добавляем элементы в список. Функция void rewiew(void) выводит список в обратном порядке в консоль для удобства функции stack. С помощью функции stack можем вынуть последний добавленный элемент списка.

**Листинг:**

**Задание 1-2:**

#include <iostream>

#include <locale.h>

typedef struct node

{

int zn;

struct node\* next;

};

struct node\* head = NULL;

struct node\* last = NULL;

struct node \*add(void)

{

int a;

struct node\* p = NULL;

p = (node\*)malloc(sizeof(struct node));

printf("Введите значение элемента:");

scanf\_s(" %d", &a);

p->zn = a;

p->next = NULL;

return p;

}

void spstore(void)

{

struct node\* p = NULL;

p = add();

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;

last = p;

}

return;

}

void rewiew(void)

{

struct node\* p = head;

while (p)

{

printf("%d ", p->zn);

p = p->next;

}

return;

}

void opr(void)

{

struct node\* p = NULL;

struct node\* str = head;

p = add();

int n,c=1;

printf("\nНомер в списке");

scanf\_s("%d", &n);

while ((c < n-1) && (str->next != NULL))

{

str = str->next;

c++;

}

p->next = str->next;

str->next = p;

return;

}

int main()

{

setlocale(LC\_ALL, "Rus");

int begin = 0;

while (begin != 4)

{

printf("\n1.Добавить элемент\n");

printf("2.Вывести набор элементов\n");

printf("3.Добавить элемент в опр место списка\n");

printf("4.Выход\n");

printf("Выбор: ");

scanf\_s("%d", &begin);

if (begin == 1)

{

spstore();

}

if (begin == 2)

{

rewiew();

}

if (begin == 3)

{

opr();

}

}

}

**Задание 3:**

#include <iostream>

#include <locale.h>

typedef struct node

{

int zn;

struct node\* next;

struct node\* prev;

};

struct node\* head = NULL;

struct node\* last = NULL;

struct node \*add(void)

{

int a;

struct node\* p = NULL;

p = (node\*)malloc(sizeof(struct node));

printf("Введите значение элемента:");

scanf\_s(" %d", &a);

p->zn = a;

p->next = head;

p->prev=NULL;

if(head) head->prev = p;

head = p;

if (!last) last=head;

return p;

}

void rewiew(void)

{

struct node\* p = head;

while (p)

{

printf("%d ", p->zn);

p = p->next;

}

return;

}

void stack(void)

{

int n=1;

struct node\* p = head;

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

{

head=p->next;

n++;

}

return;

}

int main()

{

setlocale(LC\_ALL, "Rus");

int begin = 0;

while (begin != 4)

{

printf("\n1.Добавить элемент\n");

printf("2.Вывести набор элементов\n");

printf("3.Добавить элемент в опр место списка\n");

printf("4.Выход\n");

printf("Выбор: ");

scanf\_s("%d", &begin);

if (begin == 1)

{

add();

}

if (begin == 2)

{

rewiew();

}

if (begin == 3)

{

stack();

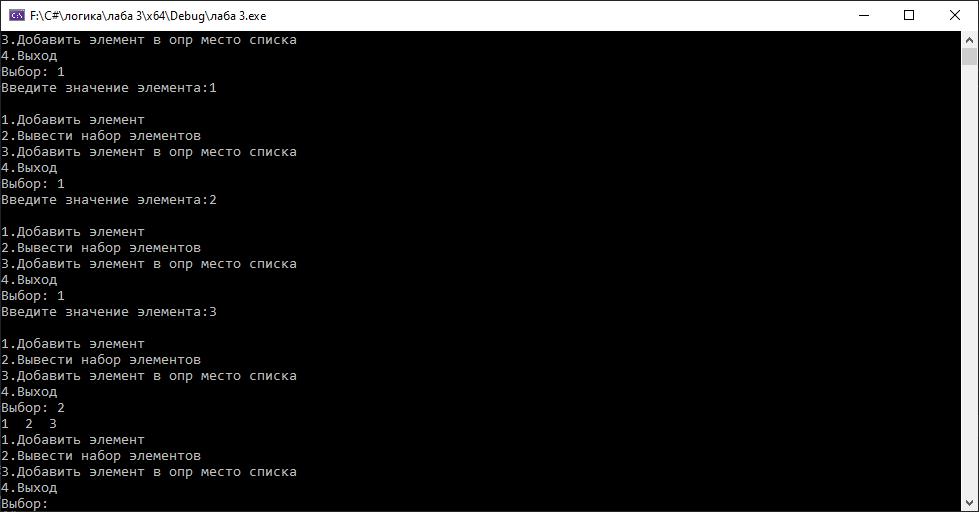
}

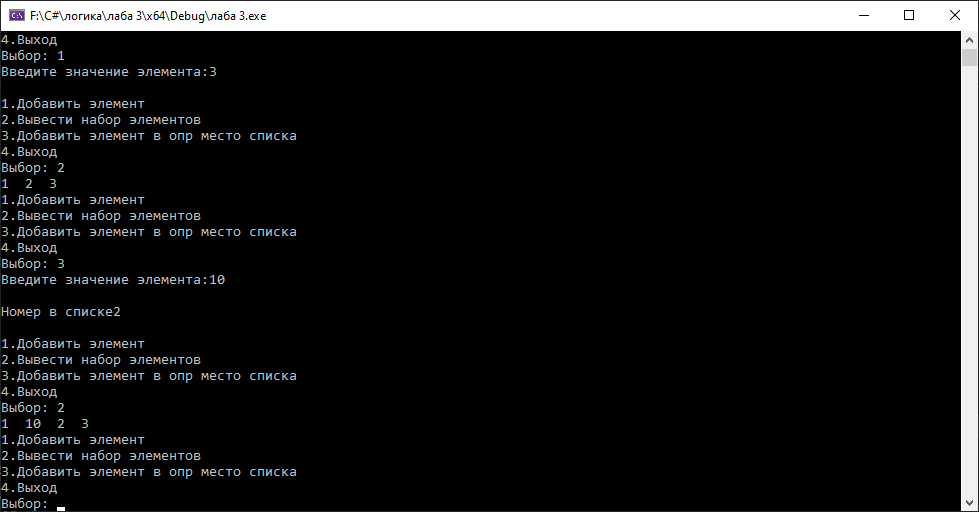
}

}

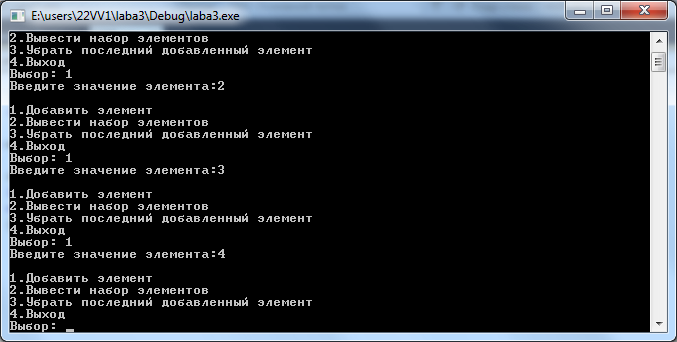
**Результаты работы программы:**

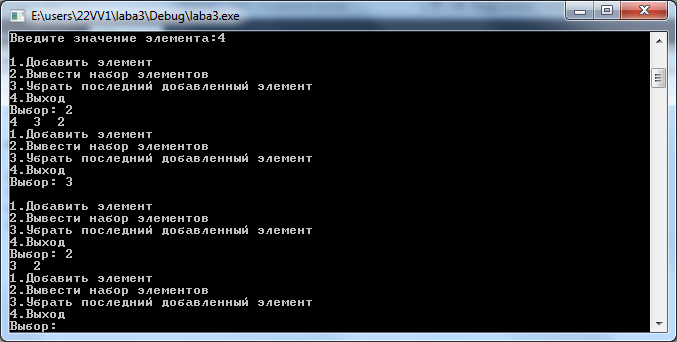
**Задание 1-2:**

****

****

**Задание 3:**

****

****

**Вывод:** в ходе выполнения лабораторной работы мы научились работать с динамическими списками. Также научились работать со структурами данных очередь и стек. Результаты работы программ совпали с результатами трассировки, следовательно программы работают без ошибок.