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 23ВВВ2

Герасимов В.Р.

Мадамкин В. М.

Приняли:

Юрова О. В.

Митрохин М. А.

Пенза 2024

**Лабораторное задание.**

**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).

**Задание 2**:

На основе приведенного кода реализуйте структуру данных *Очередь*.

**Задание 3**:

На основе приведенного кода реализуйте структуру данных *Стек.*

**Ход работы**

Задание 1:

Реализовали приоритетную очередь, код программы находится в Приложении А/lab3.1.

Результат работы программы изображен на рисунке №1:

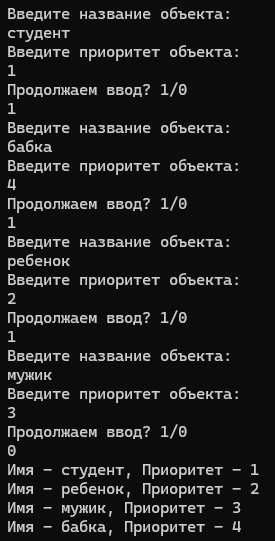


Рисунок №1

Задание 2:

Реализовали структуру данных Очередь (FIFO), код программы находится в Приложении А/lab3.2.

Результат работы программы изображен на рисунке №2:

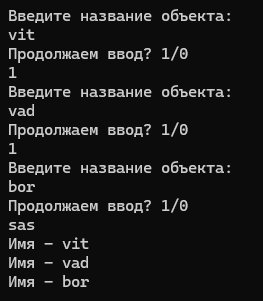


Рисунок №2

Задание 3:

Реализовали структуру данных Стек(LIFO), код программы находится в Приложении А/lab3.3.

Результат работы программы изображен на рисунке №3:

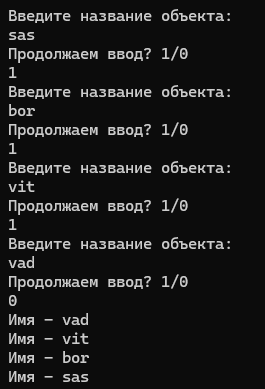


Рисунок №3

**Вывод:** реализовали приоритетную очередь, путём добавления элемента в список в соответствии с приоритетом объекта, структуру данных *Очередь*, структуру данных *Стек*.

**Приложение A**

**Lab3.1**

#define \_CRT\_SECURE\_NO\_WARNINGS

#define \_CRT\_SECURE\_NO\_DEPRECATE

#define \_CRT\_NONSTDC\_NO\_DEPRECATE

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <Windows.h>

struct node

{

char inf[256]; // полезная информация

int prior;

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

};

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

char specchar = '+';

int g = 0;

int spstore();

void review(void), del(char\* name);

char find\_el[256];

struct node\* find(char\* name); // функция нахождения элемента

struct node\* get\_struct(void); // функция создания элемента

struct node\* get\_struct(void)

{

struct node\* p = NULL;

char s[256];

int num = 0;

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

scanf("%s", s);

fflush(stdin);

if (\*s == 0)

{

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

return NULL;

}

strcpy(p->inf, s);

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

scanf("%d", &num);

fflush(stdin);

p->prior = num;

p->next = NULL;

return p; // возвращаем указатель на созданный элемент

}

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

int spstore()

{

struct node\* p = NULL;

struct node\* x = NULL;

struct node\* n = NULL;

struct node\* struc = head;

p = get\_struct();

fflush(stdin);

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

{

while (struc)

{

n = struc->next;

if ((struc->prior < p->prior) && (n == NULL)) {

struc->next = p;

last = p;

break;

}

else if ((struc->prior > p->prior)) {

p->next = struc;

if (x != NULL)

{

x->next = p;

}

if (struc == head) {

head = p;

}

break;

}

else if ((struc->prior <= p->prior)) {

x = struc;

struc = struc->next;

}

}

}

printf("Продолжаем ввод? 1/0 \n");

scanf(" %c", &specchar);

getchar();

if (specchar == '1') {

g = 1;

}

else {

g = 0;

}

return (g);

}

/\* Просмотр содержимого списка. \*/

void review(void)

{

struct node\* struc = head;

if (head == NULL)

{

printf("Список пуст\n");

}

while (struc)

{

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

struc = struc->next;

}

return;

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

spstore();

while (g == 1) {

g = spstore();

}

review();

}

**Lab3.2**

#define \_CRT\_SECURE\_NO\_WARNINGS

#define \_CRT\_SECURE\_NO\_DEPRECATE

#define \_CRT\_NONSTDC\_NO\_DEPRECATE

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <Windows.h>

struct node

{

char inf[256]; // полезная информация

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

};

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

char specchar;

int g = 0;

int spstore();

void review(void), del(char\* name);

char find\_el[256];

struct node\* find(char\* name); // функция нахождения элемента

struct node\* get\_struct(void); // функция создания элемента

struct node\* get\_struct(void)

{

struct node\* p = NULL;

char s[256];

int num = 0;

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

scanf("%s", s);

if (\*s == 0)

{

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

return NULL;

}

strcpy(p->inf, s);

p->next = NULL;

return p; // возвращаем указатель на созданный элемент

}

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

int spstore()

{

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;//голова не меняется, все в конец(FIFO)

last = p;

}

printf("Продолжаем ввод? 1/0 \n");

scanf\_s(" %c", &specchar);

getchar();

if (specchar == '1') {

g = 1;

}

else {

g = 0;

}

return (g);

}

/\* 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;

}

return;

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

spstore();

while (g == 1) {

g = spstore();

}

review();

}

**Lab3.3**

#define \_CRT\_SECURE\_NO\_WARNINGS

#define \_CRT\_SECURE\_NO\_DEPRECATE

#define \_CRT\_NONSTDC\_NO\_DEPRECATE

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <Windows.h>

struct node

{

char inf[256]; // полезная информация

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

};

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

char specchar;

int g = 0;

int spstore();

void review(void), del(char\* name);

char find\_el[256];

struct node\* find(char\* name); // функция нахождения элемента

struct node\* get\_struct(void); // функция создания элемента

struct node\* get\_struct(void)

{

struct node\* p = NULL;

char s[256];

int num = 0;

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

scanf("%s", s);

if (\*s == 0)

{

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

return NULL;

}

strcpy(p->inf, s);

p->next = NULL;

return p; // возвращаем указатель на созданный элемент

}

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

int spstore()

{

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

{

p->next = head;//постоянно меняется голова(LIFO)

head = p;

}

printf("Продолжаем ввод? 1/0 \n");

scanf\_s(" %c", &specchar);

getchar();

if (specchar == '1') {

g = 1;

}

else {

g = 0;

}

return (g);

}

/\* 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;

}

return;

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

spstore();

while (g == 1) {

g = spstore();

}

review();

}