Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра "Вычислительная техника"

Отчет

по лабораторной работе №3

по курсу “Л и ОА в ИЗ”

на тему “Динамические списки”

Выполнил студенты группы 22ВВС1:

Агапов И.Е.

Братчиков А.А.

Приняли:

Юрова О.В.

Акифьев И.В.

Пенза 2023

### Лабораторное задание

### 1. Реализовать приоритетную очередь, путём добавления элемента в список всоответствии с приоритетом объекта (т.е. объект с большим приоритетомстановится перед объектом с меньшим приоритетом).

### 2. \* На основе приведенного кода реализуйте структуру данных Очередь.

### 3. \* На основе приведенного кода реализуйте структуру данных Стек.

**Листинг**

Для задания 1

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

#include <string>

#include <iostream>

struct node

{

char inf[256]; // полезная информация

int prio;

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

};

int prior;

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

int dlinna = 0;

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

char find\_el[256];

struct node\* findprio(int prio);

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

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

int main() {

setlocale(LC\_ALL, "Russian");

printf("Введите приоритет и данные\n");

scanf("%d", &prior);

head = get\_struct();

last = head;

while (true) {

printf("Введите приоритет и данные, если хотите окончить ввод введите -1,если хотите удалить введите 0\n");

scanf("%d", &prior);

if(prior == 0){

printf("Введите удаляемый элемент\n");

scanf("%s", find\_el);

del(find\_el);

review();

} else if (prior == -1) {

printf("Ввод окончен\n");

break;

} else if(prior<0){

printf("атата нельзя так\n");

return 0;

} else {

spstore();

review();

}

}

review();

return 0;

}

// Функции добавления элемента, просмотра списка

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);

}

scanf("%s", s);

p->prio=prior ;

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

}

void spstore(void)

{

struct node\* ideal = NULL;

struct node\* p = NULL;

p = get\_struct();

if (head->prio <= p->prio)

{

p->next = head;

head = p;

}

else if (head->prio > p->prio)

{

ideal =findprio(p->prio);

if (ideal == NULL) {

last->next = p;

last = p;

}

else {

printf("%s\n",ideal->inf);

p->next = ideal->next;

ideal->next = p;

}

}

return;

}

/\* 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->prio);

struc = struc->next;

}

return;

}

struct node\* findprio(int prio)

{

struct node\* struc = head;

struct node\* temp =NULL;

if (head == NULL)

{

printf("Список пуст\n");

}

while (struc)

{

if (prio >= struc->prio)

{

return temp;

}

temp = struc;

struc = struc->next;

}

return NULL;

}

void del(char \*name)

{

struct node \*struc = head; // указатель, проходящий по списку установлен на начало

struct node \*prev;// указатель на предшествующий удаляемому элемент

int flag = 0; // индикатор отсутствия удаляемого элемента в списке

if (head == NULL) // если голова списка равна NULL, то список пуст

{

printf("Список пуст\n");

return;

}

if (strcmp(name, struc->inf) == 0) // если удаляемый элемент - первый

{

flag = 1;

head = struc->next; // установливаем голову на следующий элемент

free(struc); // удаляем первый элемент

struc = head; // устанавливаем указатель для продолжения поиска

}

else

{

prev = struc;

struc = struc->next;

}

while (struc) // проход по списку и поиск удаляемого элемента

{

if (strcmp(name, struc->inf) == 0) // если нашли, то

{

flag = 1; // выставляем индикатор

if (struc->next) // если найденный элемент не последний в списке

{

prev->next = struc->next; // меняем указатели

free(struc); // удаляем элемент

struc = prev->next; // устанавливаем указатель для продолжения

}

else // если найденный элемент последний в списке

{

prev->next = NULL; // обнуляем указатель предшествующего

free(struc); // удаляем элемент

return;

}

}

else

{

prev = struc; // устанавливаем указатели для продолжения поиска

struc = struc->next;

}

}

if (flag == 0) // если флаг = 0, значит нужный элемент не

{

printf("Элемент не найден\n");

return;

}

}

struct node \*find(char \*name)

{

struct node \*struc = head;

if (head == NULL)

{

printf("Список пуст\n");

}

while (struc)

{

if (strcmp(name, struc->inf) == 0)

{

return struc;

}

struc = struc->next;

}

printf("Элемент не найден\n");

return NULL;

}

Для задания 2

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

#include <string>

#include <iostream>

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

int dlinna = 0;

void 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

void main() {

setlocale(LC\_ALL, "Russian");

printf("Введите данные\n");

head = get\_struct();

last = head;

while (true) {

printf("Введите данные, если хотите окончить ввод введите \*\n");

spstore();

if (f == NULL) {

break;

}

}

review();

printf("Введите данные, которые хотите удалить\n");

scanf("%s", find\_el);

del(find\_el);

review();

}

void spstore(void)

{

struct node\* p = NULL;

p = get\_struct();

if (p == NULL) {

f = NULL;

}

last->next = p;

last = p;

f = p;

return;

}

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);

}

scanf("%s", s);

if (strcmp(s, "\*") == 0)

{

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

return NULL;

}

strcpy(p->inf, s);

p->next = NULL;

return p; // возвращаем указатель на созданный элемент

}

/\* 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;

}

struct node\* find(char\* name)

{

struct node\* struc = head;

if (head == NULL)

{

printf("Список пуст\n");

}

while (struc)

{

if (strcmp(name, struc->inf) == 0)

{

return struc;

}

struc = struc->next;

}

printf("Элемент не найден\n");

return NULL;

}

void del(char\* name)

{

struct node\* struc = head; // указатель, проходящий по списку установлен на начало

struct node\* prev=NULL;// указатель на предшествующий удаляемому элемент

int flag = 0; // индикатор отсутствия удаляемого элемента в списке

if (head == NULL) // если голова списка равна NULL, то список пуст

{

printf("Список пуст\n");

return;

}

if (strcmp(name, struc->inf) == 0) // если удаляемый элемент - первый

{

flag = 1;

head = struc->next; // установливаем голову на следующий элемент

free(struc); // удаляем первый элемент

struc = head; // устанавливаем указатель для продолжения поиска

}

else

{

prev = struc;

struc = struc->next;

}

while (struc) // проход по списку и поиск удаляемого элемента

{

if (strcmp(name, struc->inf) == 0) // если нашли, то

{

flag = 1; // выставляем индикатор

if (struc->next) // если найденный элемент не последний в списке

{

prev->next = struc->next; // меняем указатели

free(struc); // удаляем элемент

struc = prev->next; // устанавливаем указатель для продолжения

}

else // если найденный элемент последний в списке

{

prev->next = NULL; // обнуляем указатель предшествующего

free(struc); // удаляем элемент

return;

}

}

else

{

prev = struc; // устанавливаем указатели для продолжения поиска

struc = struc->next;

}

}

if (flag == 0) // если флаг = 0, значит нужный элемент не

{

printf("Элемент не найден\n");

return;

}

}

Для задания 3

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

#include <string>

#include <iostream>

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

int dlinna = 0;

void 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

void main() {

setlocale(LC\_ALL, "Russian");

printf("Введите данные\n");

head = get\_struct();

last = head;

while (true) {

printf("Введите данные, если хотите окончить ввод введите \*\n");

spstore();

if (f == NULL) {

break;

}

}

review();

printf("Введите данные, которые хотите удалить\n");

scanf("%s", find\_el);

del(find\_el);

review();

}

void spstore(void)

{

struct node\* p = NULL;

p = get\_struct();

if (p == NULL) {

f = NULL;

}

else {

p->next = last;

last = p;

head = last;

}

f = p;

return;

}

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);

}

scanf("%s", s);

if (strcmp(s, "\*") == 0)

{

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

return NULL;

}

strcpy(p->inf, s);

p->next = NULL;

return p; // возвращаем указатель на созданный элемент

}

/\* 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;

}

struct node\* find(char\* name)

{

struct node\* struc = head;

if (head == NULL)

{

printf("Список пуст\n");

}

while (struc)

{

if (strcmp(name, struc->inf) == 0)

{

return struc;

}

struc = struc->next;

}

printf("Элемент не найден\n");

return NULL;

}

void del(char\* name)

{

struct node\* struc = head; // указатель, проходящий по списку установлен на начало

struct node\* prev=NULL;

int flag = 0; // индикатор отсутствия удаляемого элемента в списке

if (strcmp(name, struc->inf) == 0) // если удаляемый элемент - первый

{

flag = 1;

head = struc->next; // установливаем голову на следующий элемент

free(struc); // удаляем первый элемент

struc = head; // устанавливаем указатель для продолжения поиска

return;

}

prev=struc;

struc=struc->next;

while (struc) {

if (strcmp(name, struc->inf) == 0){

flag=1;

prev->next=struc->next;

free(struc);

return;

}

}

if (flag == 0) // если флаг = 0, значит нужный элемент не

{

printf("Элемент не найден\n");

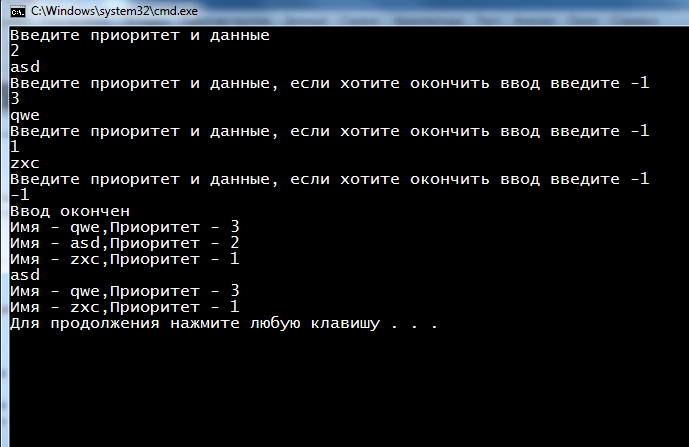
return;

}

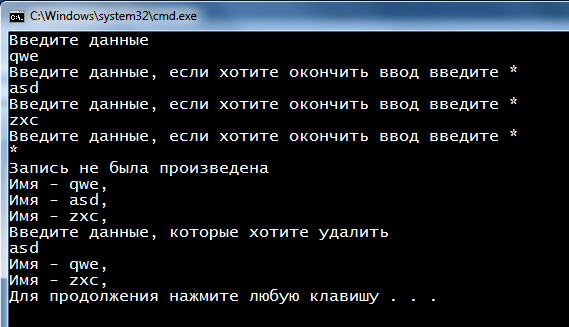
}

**Результаты работ программ**

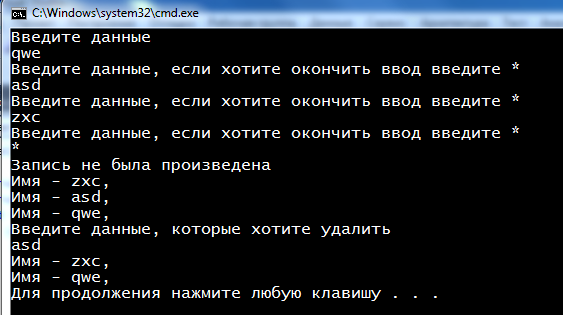
Задание 1



Задание 2



Задание 3



**Вывод**

### В ходе выполнения лабораторной работы были обобщены знания о структурах данных. Были реализованы такие структуры данных как: приоритетная очередь, очередь, в которой первое вошедшее значение выходит первым(FIFO), стэк, в котором первое вошедшее значение выходит последним(FILO).

### Для каждой из этих структур были разработаны такие функции как: добавление, поиск, удаление элемента, вывод всей структуры.