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 22ВВВ2:

Юртаев Д. Ю.

Шатин Д. Д.

Харитонов А. А.

Приняли:

Акифьев И. В.

Митрохин М. А.

Пенза 2023

**Цель работы**

Повторить и усовершенствовать полученные ранее знания о динамических списках на языке СИ.

**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.

**Листинг**

**Lab3.1**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <Windows.h>

struct node

{

char inf[256]; // полезная информация

int priority;

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;

// Функции добавления элемента, просмотра списка

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

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int choice = 1;

while (choice)

{

printf("Выберите действие:\n");

printf("(1) Просмотр списка\n");

printf("(2) Добавление элемента в список\n");

printf("(3) Удаление элемента из списка\n");

printf("(4) Поиск элемента в списке\n");

printf("(0) Выход из программы\n\n");

printf("Ваш выбор: ");

scanf("%d", &choice);

system("cls");

switch (choice)

{

case 1:

review();

system("pause");

system("cls");

break;

case 2:

spstore();

system("pause");

system("cls");

break;

case 3:

printf("Введите содержимое элемента\n>> ");

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

del(find\_el);

system("pause");

system("cls");

break;

case 4:

printf("Введите содержимое элемента\n>> ");

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

find(find\_el);

system("pause");

system("cls");

break;

}

}

return 0;

}

struct node\* get\_struct(void)

{

struct node\* p = NULL;

char s[256];

int prior;

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);

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

scanf("%d", &prior);

if (\*s == 0 || !prior)

{

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

return NULL;

}

strcpy(p->inf, s);

p->priority = prior;

p->next = NULL;

return p; // возвращаем указатель на созданный элемент

}

/\* Добавление в список элемента \*/

void spstore(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->priority > p->priority) && p != NULL) // если добавляем перед head

{

p->next = head;

head = p;

last = p->next;

}

else if ((p->priority > last->priority) && p != NULL) // если новый элемент добавляется после last

{

last->next = p;

last = p;

}

else if (head != NULL && p != NULL)

{

if (head->next == NULL && p != NULL) // если элемент один в списке, и нужно добавить после него

{

head->next = p;

last = p;

return;

}

struct node\* prev = head; // элемент, после которого будет вставлен новый

struct node\* struc = prev->next;

while (struc)

{

if (struc->priority > p->priority)

{

p->next = struc;

prev->next = p;

}

prev = struc;

struc = struc->next;

}

}

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->priority);

//printf("Имя - %s, \n", struc->inf);

struc = struc->next;

}

// Задание 1. Очередь с приоритетным исключение

/\*for (int p = 0; p < 5; p++)

{

struc = head;

while (struc)

{

if (struc->priority == p + 1)

{

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

}

struc = struc->next;

}

}

\*/

return;

}

/\* Поиск элемента по содержимому. \*/

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;

}

/\* Удаление элемента по содержимому. \*/

void del(char\* name)

{

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

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 поиска

}

else // если найденный элемент последний в списке

{

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

free(struc); // удаляем элемент

last = prev;

return;

}

}

else // если не нашли, то

{

prev = struc; // устанавливаем указатели для продолжения поиска

struc = struc->next;

}

}

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

{

printf("Элемент не найден\n");

return;

}

}

**Lab3.2**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <Windows.h>

// Задание 2. Реализация очереди

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;

// Функции добавления элемента, просмотра списка

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

char find\_el[256];

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

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

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int choice = 1;

int choice4 = 1;

while (choice)

{

printf("Длина списка - %d\n\n", dlinna);

printf("Выберите действие:\n");

printf("(1) Просмотр списка\n");

printf("(2) Добавление элемента в список\n");

printf("(3) Удаление любого элемента из списка\n");

printf("(4) Извлечение первого элемента из списка\n");

printf("(5) Поиск элемента в списке\n");

printf("(0) Выход из программы\n\n");

printf("Ваш выбор: ");

scanf("%d", &choice);

system("cls");

switch (choice)

{

case 1:

review();

system("pause");

system("cls");

break;

case 2:

spstore();

system("pause");

system("cls");

break;

case 3:

printf("Введите содержимое элемента\n>> ");

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

del(find\_el);

system("pause");

system("cls");

break;

case 4:

printf("Выберите действие:\n");

printf("(1) Извлечь с сохранением элемента\n");

printf("(2) Извлечь с удалением элемента\n\n");

printf("Ваш выбор: ");

scanf("%d", &choice4);

system("cls");

pop(choice4);

system("pause");

system("cls");

break;

case 5:

printf("Введите содержимое элемента\n>> ");

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

find(find\_el);

system("pause");

system("cls");

break;

}

}

return 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

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

}

/\* Добавление в список элемента \*/

void spstore(void)

{

struct node\* p = NULL;

p = get\_struct();

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

{

head = p;

last = p;

dlinna += 1;

}

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

{

last->next = p;

last = p;

dlinna += 1;

}

return;

}

/\* 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;

}

/\* Поиск элемента по содержимому. \*/

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;

}

/\* Удаление элемента по содержимому. \*/

void del(char\* name)

{

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

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

dlinna -= 1;

}

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 поиска

dlinna -= 1;

}

else // если найденный элемент последний в списке

{

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

free(struc); // удаляем элемент

last = prev;

dlinna -= 1;

return;

}

}

else // если не нашли, то

{

prev = struc; // устанавливаем указатели для продолжения поиска

struc = struc->next;

}

}

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

{

printf("Элемент не найден\n");

return;

}

}

void pop(int choice)

{

struct node\* struc = head;

if (choice && choice < 3)

{

if (struc == NULL)

{

printf("Список пуст\n");

return;

}

if (choice == 1)

{

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

}

else if (choice == 2)

{

head = struc->next;

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

free(struc);

dlinna -= 1;

}

}

else

{

printf("Не правильный пункт меню!\n");

}

}

**Lab3.3**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <Windows.h>

// Задание 3. Реализация стека

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;

// Функции добавления элемента, просмотра списка

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

char find\_el[256];

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

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

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int choice = 1;

int choice4 = 1;

while (choice)

{

printf("Длина списка - %d\n\n", dlinna);

printf("Выберите действие:\n");

printf("(1) Просмотр списка\n");

printf("(2) Добавление элемента в список\n");

printf("(3) Удаление любого элемента из списка\n");

printf("(4) Извлечение первого элемента из списка\n");

printf("(5) Поиск элемента в списке\n");

printf("(0) Выход из программы\n\n");

printf("Ваш выбор: ");

scanf("%d", &choice);

system("cls");

switch (choice)

{

case 1:

review();

system("pause");

system("cls");

break;

case 2:

spstore();

system("pause");

system("cls");

break;

case 3:

printf("Введите содержимое элемента\n>> ");

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

del(find\_el);

system("pause");

system("cls");

break;

case 4:

printf("Выберите действие:\n");

printf("(1) Извлечь с сохранением элемента\n");

printf("(2) Извлечь с удалением элемента\n\n");

printf("Ваш выбор: ");

scanf("%d", &choice4);

system("cls");

pop(choice4);

system("pause");

system("cls");

break;

case 5:

printf("Введите содержимое элемента\n>> ");

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

find(find\_el);

system("pause");

system("cls");

break;

}

}

return 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

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

}

/\* Добавление в список элемента \*/

void spstore(void)

{

struct node\* p = NULL;

p = get\_struct();

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

{

head = p;

last = p;

dlinna += 1;

}

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

{

p->next = head;

head = p;

dlinna += 1;

}

return;

}

/\* 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;

}

/\* Поиск элемента по содержимому. \*/

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;

}

/\* Удаление элемента по содержимому. \*/

void del(char\* name)

{

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

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

dlinna -= 1;

}

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 поиска

dlinna -= 1;

}

else // если найденный элемент последний в списке

{

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

free(struc); // удаляем элемент

last = prev;

dlinna -= 1;

return;

}

}

else // если не нашли, то

{

prev = struc; // устанавливаем указатели для продолжения поиска

struc = struc->next;

}

}

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

{

printf("Элемент не найден\n");

return;

}

}

void pop(int choice)

{

struct node\* struc = head;

if (choice && choice < 3)

{

if (struc == NULL)

{

printf("Список пуст\n");

return;

}

if (choice == 1)

{

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

}

else if (choice == 2)

{

head = struc->next;

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

free(struc);

dlinna -= 1;

}

}

else

{

printf("Не правильный пункт меню!\n");

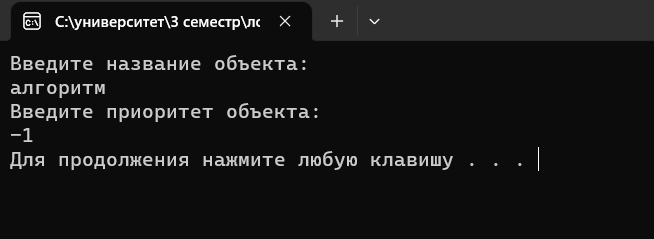
}

}

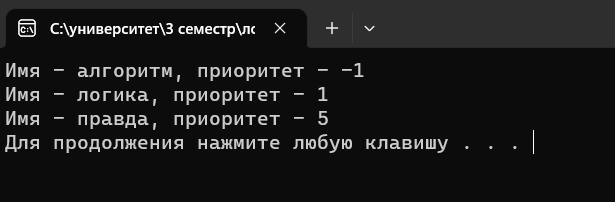
**Результаты работы программ**

**Очередь с приоритетом**

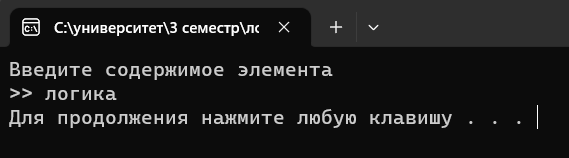
Добавление элементов

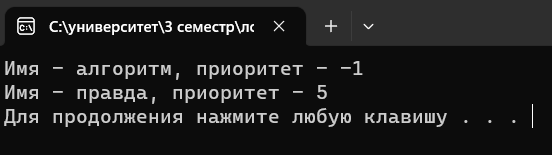


Просмотр списка



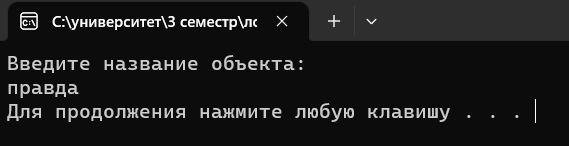
Удаление элементов



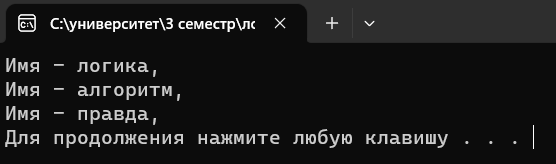


**Очередь**

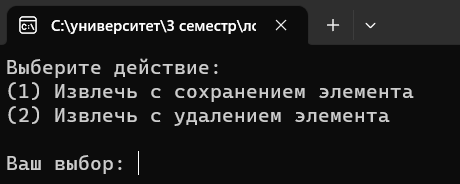
Добавление элементов

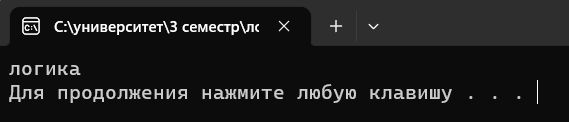


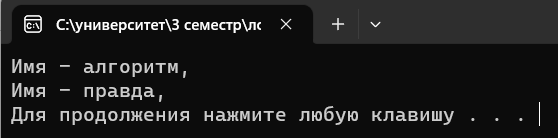
Просмотр списка



Извлечение элементов из очереди

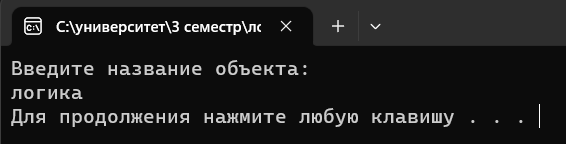




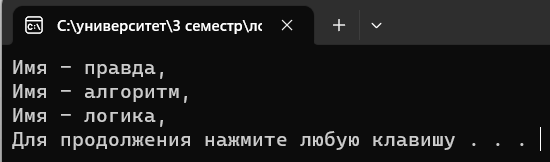


**Стек**

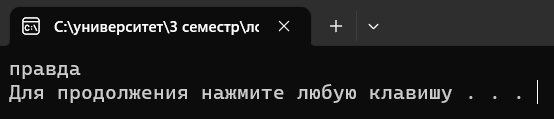
Добавление элементов в стек

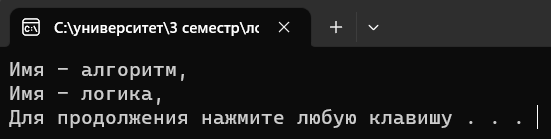


Просмотр списка



Извлечение элементов из стека





**Вывод**

В ходе работы, удалось повторить и усовершенствовать полученные ранее знания о динамических списках на языке СИ.