Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №3

по теме «Динамические списки»

Выполнили:

Студенты группы 21ВВ2

Кривенкова В.С.

Козлова К.С.

Принял:

Митрохин М.А.

Юрова О.В.

Пенза 2022

**Цель работы:**

Целью работы является получение навыков организации данных и их обработки в динамической памяти.

**Лабораторные работы:**

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.

**Результат работы программы:**



Рисунок 1: 1 задание

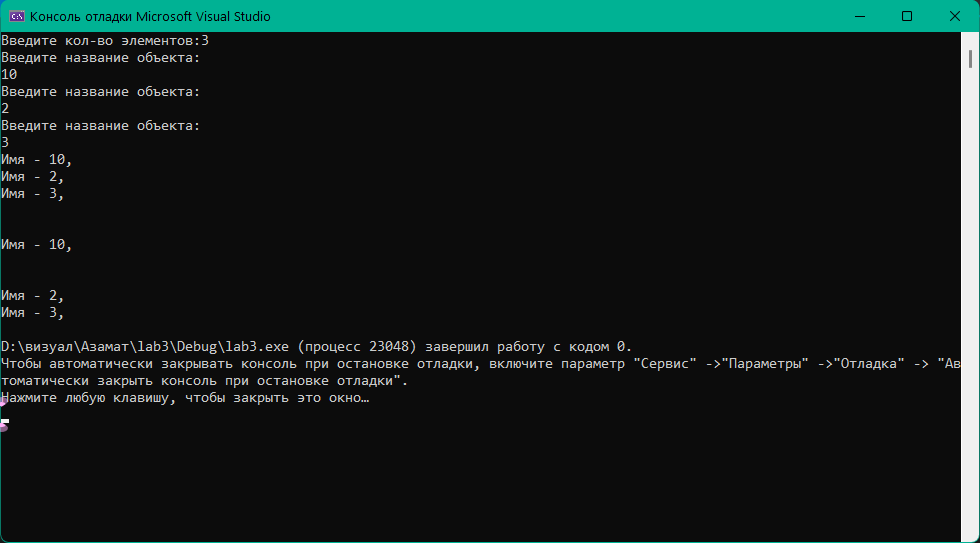


Рисунок 2: 2 задание

Принцип работы очереди заключается в том , что элементы добавляются в конец списка, а удаляются из начала.

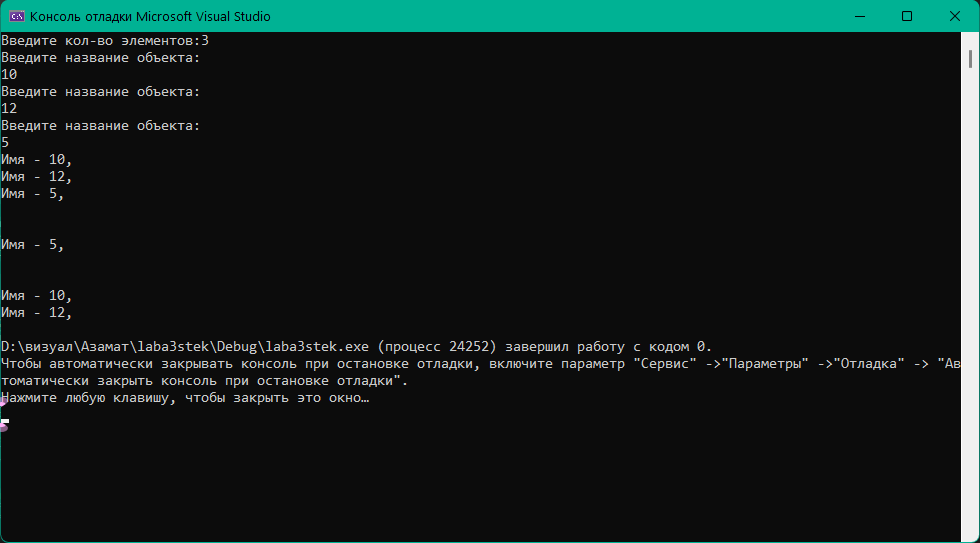


Рисунок 3: 3 задание

Принцип работы стека заключается в том , что элементы добавляются в конец списка и удаляются из конца.

**ЛИСТИНГ**

**Задание 2**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <iostream>

#include <stdlib.h>

#include <locale.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

int dlinna = 0;

// Функции добавления элемента, просмотра списка

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

void pop\_front(void);

char find\_el[256];

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

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

int main()

{

setlocale(0, "");

int n;

printf("Введите кол-во элементов:");

scanf("%d", &n);

for (int k = 1; k <= n; k++)

{

spstore();

};

review();

printf("\n\n");

pop\_front();

printf("\n\n");

review();

return 0;

}

void pop\_front(void)

{

struct node\* tmp = head;

head = tmp->next;

printf("Имя - %s, \n", tmp->inf);

free(tmp);

}

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

}

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

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 != NULL && p != NULL) // список уже есть, то вставляем в конец

{

last->next = p;

last = p;

}

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;

}

**Задание 1**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <iostream>

#include <stdlib.h>

#include <locale.h>

struct node

{

char inf[256]; // полезная информация

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

char prior[256];

};

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);

void pop\_front(void);

char find\_el[256];

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

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

int main()

{

setlocale(0, "");

int n;

printf("Введите кол-во элементов:");

scanf("%d", &n);

for (int k = 1; k <= n; k++)

{

spstore();

};

review();

printf("\n\n");

pop\_front();

printf("\n\n");

review();

return 0;

}

void pop\_front(void)

{

struct node\* tmp = head;

head = tmp->next;

printf("Имя - %s, \n", tmp->inf);

free(tmp);

}

struct node\* get\_struct(void)

{

struct node\* p = NULL;

char s[256];

char pr[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);

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

scanf("%s", pr);

if (\*s == 0)

{

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

return NULL;

} \

strcpy(p->inf, s);

strcpy(p->prior, pr);

p->next = NULL;

return p; // возвращаем указатель на созданный элемент

}

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

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;

return;

}

struct node\* tmp = head;

struct node\* prev = tmp;

while (tmp != NULL)

{

if (atoi(p->prior) >= atoi(tmp->prior))

{

break;

}

else

{

prev = tmp;

tmp = tmp->next;

}

}

if (tmp==head)

{

p->next = head;

head = p;

}

else if (tmp == NULL)

{

last->next = p;

last = p;

}

else

{

p->next = prev->next;

prev->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, \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;

}

**Задание 3**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <iostream>

#include <stdlib.h>

#include <locale.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

int dlinna = 0;

// Функции добавления элемента, просмотра списка

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

void pop\_back(void);

char find\_el[256];

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

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

int main()

{

setlocale(0, "");

int n;

printf("Введите кол-во элементов:");

scanf("%d", &n);

for (int k = 1; k <= n; k++)

{

spstore();

};

review();

printf("\n\n");

pop\_back();

printf("\n\n");

review();

return 0;

}

void pop\_back(void)

{

struct node\* tmp = head;

struct node\* prev = tmp;

while (tmp->next != NULL)

{

prev = tmp;

tmp = tmp->next;

}

printf("Имя - %s, \n", tmp->inf);

prev->next = NULL;

free(tmp);

}

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

}

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

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 != NULL && p != NULL) // список уже есть, то вставляем в конец

{

last->next = p;

last = p;

}

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;

}

**ВЫВОД:**

Мы получили навыки организации данных и их обработки в динамической памяти.