Пензенский государственный университет

Кафедра «Вычислительная техника»

ОТЧЕТ

по лабораторной работе № 3

по дисциплине: "Логика и основы алгоритмизации в инженерных задачах"

на тему: "Динамические списки"

Выполнили студенты гр. 22ВВП2:

Корнилов В.М.

Самофалова А.В.

Горбунов Д.А.

Приняли:

Акифьев И.В.

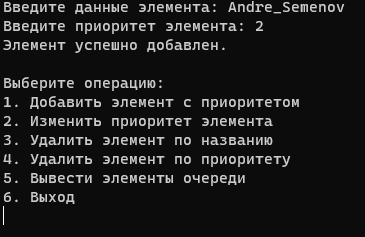
Юрова О.В.

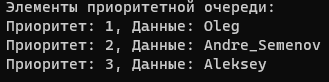
Пенза, 2023

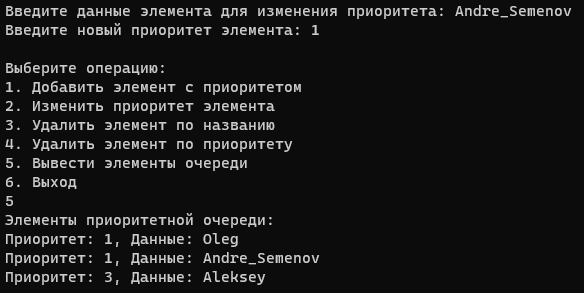
**Цель:** научиться создавать Приоритетную Очередь. Научиться создавать структуру данных – Стек. Научиться создавать структуру данных – Очередь.

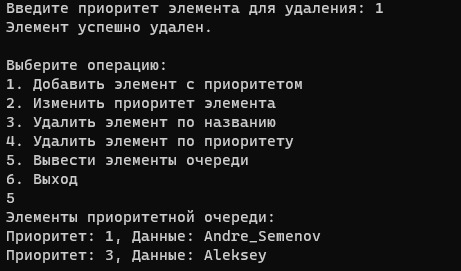
**Задание**

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).

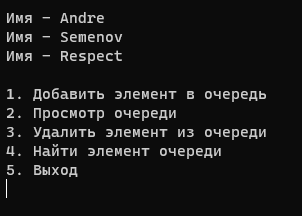






****

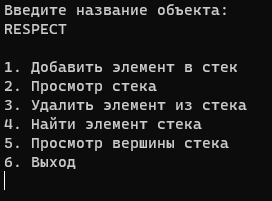
1. \* На основе приведенного кода реализуйте структуру данных *Очередь*.

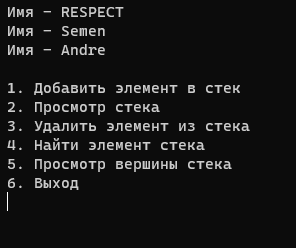


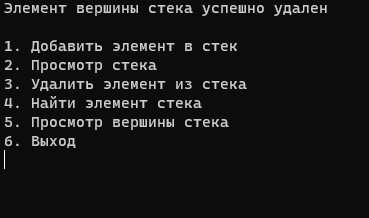


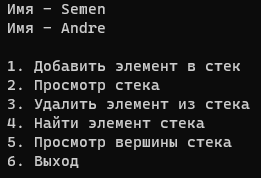


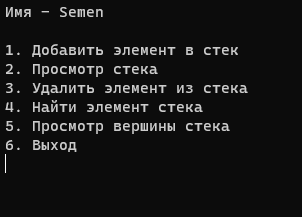
1. \* На основе приведенного кода реализуйте структуру данных *Стек*.











**Вывод:**

Научились создавать Приоритетную Очередь. Научились создавать структуру данных – Стек. Научились создавать структуру данных – Очередь.

**Листинг**

**Приложение А (Код программы 1)**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <windows.h>

#include <locale.h>

typedef struct Node {

int priority;

char data[100];

struct Node\* next;

} Node;

Node\* createNode(int priority, char\* data) {

Node\* newNode = (Node\*)malloc(sizeof(Node));

newNode->priority = priority;

strcpy(newNode->data, data);

newNode->next = NULL;

return newNode;

}

void enqueue(Node\*\* head, int priority, char\* data) {

Node\* newNode = createNode(priority, data);

if (\*head == NULL || priority < (\*head)->priority) {

newNode->next = \*head;

\*head = newNode;

}

else {

Node\* temp = \*head;

while (temp->next != NULL && temp->next->priority <= priority) {

temp = temp->next;

}

newNode->next = temp->next;

temp->next = newNode;

}

}

void dequeue(Node\*\* head) {

if (\*head == NULL) {

printf("Очередь пуста.\n");

return;

}

Node\* temp = \*head;

\*head = (\*head)->next;

free(temp);

}

void changePriority(Node\*\* head, char\* data, int newPriority) {

if (\*head == NULL) {

printf("Очередь пуста.\n");

return;

}

Node\* temp = \*head;

while (temp != NULL) {

if (strcmp(temp->data, data) == 0) {

temp->priority = newPriority;

return;

}

temp = temp->next;

}

printf("Элемент не найден.\n");

}

void printQueue(Node\* head) {

if (head == NULL) {

printf("Очередь пуста.\n");

return;

}

printf("Элементы приоритетной очереди:\n");

while (head != NULL) {

printf("Приоритет: %d, Данные: %s\n", head->priority, head->data);

head = head->next;

}

}

void deleteByPriority(Node\*\* head, int priority) {

if (\*head == NULL) {

printf("Очередь пуста.\n");

return;

}

Node\* temp = \*head;

Node\* prev = NULL;

if (temp != NULL && temp->priority == priority) {

\*head = temp->next;

free(temp);

printf("Элемент успешно удален.\n");

return;

}

while (temp != NULL && temp->priority != priority) {

prev = temp;

temp = temp->next;

}

if (temp == NULL) {

printf("Элемент с указанным приоритетом не найден.\n");

return;

}

prev->next = temp->next;

free(temp);

printf("Элемент успешно удален.\n");

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Node\* head = NULL;

int choice;

while (1) {

printf("\nВыберите операцию:\n");

printf("1. Добавить элемент с приоритетом\n");

printf("2. Изменить приоритет элемента\n");

printf("3. Удалить элемент по названию\n");

printf("4. Удалить элемент по приоритету\n");

printf("5. Вывести элементы очереди\n");

printf("6. Выход\n");

scanf("%d", &choice);

char data[100];

int priority;

switch (choice) {

case 1:

printf("Введите данные элемента: ");

scanf("%s", data);

printf("Введите приоритет элемента: ");

scanf("%d", &priority);

enqueue(&head, priority, data);

printf("Элемент успешно добавлен.\n");

break;

case 2:

printf("Введите данные элемента для изменения приоритета: ");

scanf("%s", data);

printf("Введите новый приоритет элемента: ");

scanf("%d", &priority);

changePriority(&head, data, priority);

break;

case 3:

printf("Введите данные элемента для удаления: ");

scanf("%s", data);

dequeue(&head);

printf("Элемент успешно удален.\n");

break;

case 4:

printf("Введите приоритет элемента для удаления: ");

scanf("%d", &priority);

deleteByPriority(&head, priority);

break;

case 5:

printQueue(head);

break;

case 6:

exit(0);

default:

printf("Неверный выбор.\n");

}

}

return 0;

}**Приложение Б (Код программы 2)**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <locale.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

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

struct node\* get\_struct(void)

{

struct node\* p = NULL;

char s[256];

if ((p = (struct node\*)malloc(sizeof(struct node))) == NULL) // выделяем память под новый элемент

{

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

}

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;

}

void review(void)

{

struct node\* struc = head;

if (head == NULL)

{

printf("Очередь пуста\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("Очередь пуста\n");

}

while (struc)

{

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

{

printf("Элемент %s найден", name);

return struc;

}

struc = struc->next;

}

printf("Элемент не найден\n");

return NULL;

}

/\* Удаление элемента по содержимому. \*/

void del(char\* name)

{

struct node\* struc = head;

struct node\* prev = NULL;

int flag = 0;

if (head == NULL)

{

printf("Очередь пуста\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

printf("Элемент %s удален", name);

struc = head; // устанавливаем указатель для продолжения поиска

}

else

{

prev = struc;

struc = struc->next;

}

while (struc)

{

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

{

flag = 1; // выставляем индикатор

if (struc->next)

{

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

free(struc); // удаляем элемент

printf("Элемент %s удален", name);

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

}

else

{

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

free(struc); // удаляем элемент

printf("Элемент %s удален", name);

return;

}

}

else

{

prev = struc; // устанавливаем указатели для продолжения поиска

struc = struc->next;

}

}

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

{

printf("Элемент %s не найден\n", name);

return;

}

}

int menu() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int cc;

char d[256], f[256];

while (1) {

printf("\n1. Добавить элемент в очередь\n");

printf("2. Просмотр очереди\n");

printf("3. Удалить элемент из очереди\n");

printf("4. Найти элемент очереди\n");

printf("5. Выход\n");

scanf("%d", &cc);

switch (cc) {

case 1:

system("cls");

spstore();

break;

case 2:

system("cls");

review();

break;

case 3:

system("cls");

printf("Введите название элемента\n");

scanf("%s", d);

del(d);

break;

case 4:

system("cls");

printf("Введите название элемента\n");

scanf("%s", f);

find(f);

break;

case 5:

exit(1);

break;

}

}

return 0;

}

int main() {

menu();

}

**Приложение С (Код программы 3)**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <locale.h>

#include <Windows.h>

struct node

{

char inf[256]; // полезная информация

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

};

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

// Функции добавления элемента, просмотра стека

void spstore(void), review(void);

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];

if ((p = (struct node\*)malloc(sizeof(struct node))) == NULL) // выделяем память под новый элемент стека

{

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

}

void spstore(void)

{

struct node\* p = NULL;

p = get\_struct();

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

{

head = p;

last = p;

}

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

{

p->next = last;

last = p;

}

return;

}

/\* Просмотр содержимого стека. \*/

void review(void)

{

struct node\* struc = last;

if (head == NULL)

{

printf("Стек пуст\n");

}

while (struc)

{

printf("Имя - %s \n", struc->inf);

struc = struc->next;

}

return;

}

void review\_head(void)

{

struct node\* struc = last;

if (head == NULL)

{

printf("Стек пуст\n");

}

printf("Имя - %s \n", struc->inf);

return;

}

/\* Поиск элемента по содержимому. \*/

struct node\* find(char\* name)

{

struct node\* struc = last;

if (head == NULL)

{

printf("Стек пуст\n");

}

while (struc)

{

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

{

printf("Элемент %s найден", name);

return struc;

}

struc = struc->next;

}

printf("Элемент не найден\n");

return NULL;

}

void del\_head(void)

{

struct node\* struc = last;

struct node\* prev = NULL;

if (head == NULL)

{

printf("Список пуст\n");

}

prev = last->next;

free(last);

last = prev;

printf("Элемент вершины стека успешно удален\n");

return;

}

int menu() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int cc;

char d[256], f[256];

while (1) {

printf("\n1. Добавить элемент в стек\n");

printf("2. Просмотр стека\n");

printf("3. Удалить элемент из стека\n");

printf("4. Найти элемент стека\n");

printf("5. Просмотр вершины стека\n");

printf("6. Выход\n");

scanf("%d", &cc);

switch (cc) {

case 1:

system("cls");

spstore();

break;

case 2:

system("cls");

review();

break;

case 3:

system("cls");

del\_head();

break;

case 4:

system("cls");

printf("Введите название элемента\n");

scanf("%s", f);

find(f);

break;

case 5:

system("cls");

review\_head();

break;

case 6:

exit(1);

break;

}

}

return 0;

}

int main() {

menu();

}