Министерство науки и высшего образования РФ

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 22ВВП2:

Перкин П. О.

Широкова И. Д.

Приняли

Акифьев И.В.

Юрова О.В.

Пенза 2023

**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**

**Задание 1**:

**#define \_CRT\_SECURE\_NO\_WARNINGS**

**#include <locale.h>**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <string.h>**

**struct node {**

**char inf[256];**

**int priority;**

**struct node\* next;**

**};**

**struct node\* head = NULL, \* last = NULL, \* f = NULL;**

**int dlinna = 0;**

**struct node\* get\_struct(void) {**

**struct node\* p = NULL;**

**char s[256];**

**int priority;**

**if ((p = (struct node\*)malloc(sizeof(struct node))) ==**

**NULL) {**

**printf("Ошибка при распределении памяти\n");**

**exit(1);**

**}**

**printf("Введите название объекта: \n");**

**scanf("%s", s);**

**if (\*s == 0) {**

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

**return NULL;**

**}**

**strcpy(p->inf, s);**

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

**scanf("%d", &priority);**

**p->priority = priority;**

**p->next = NULL;**

**return p;**

**}**

**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) {**

**struct node\* current = head;**

**struct node\* prev = NULL;**

**while (current != NULL && p->priority >= current -> priority) {**

**prev = current;**

**current = current->next;**

**}**

**if (prev == NULL) {**

**p->next = head;**

**head = p;**

**}**

**else {**

**prev->next = p;**

**p->next = current;**

**if (current == NULL) {**

**last = p;**

**}**

**}**

**}**

**}**

**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);**

**struc = struc->next;**

**}**

**}**

**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;**

**struct node\* prev = NULL;**

**int flag = 0;**

**if (head == NULL)**

**{**

**printf("Список пуст\n");**

**return;**

**}**

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

**{**

**flag = 1;**

**head = struc->next;**

**free(struc);**

**struc = head;**

**}**

**else {**

**prev = struc;**

**struc = struc->next;**

**}**

**while (struc)**

**{**

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

**{**

**flag = 1;**

**if (struc->next)**

**{**

**prev->next = struc->next;**

**free(struc);**

**struc = prev->next;**

**}**

**else**

**{**

**prev->next = NULL;**

**free(struc);**

**last = prev;**

**return;**

**}**

**}**

**else**

**{**

**prev = struc;**

**struc = struc->next;**

**}**

**}**

**if (flag == 0)**

**{**

**printf("Элемент не найден\n");**

**return;**

**}**

**}**

**int main() {**

**setlocale(LC\_ALL, "Rus");**

**int choice;**

**char name[256];**

**struct node\* found = NULL;**

**while (1) {**

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

**printf("2. Просмотреть очередь\n");**

**printf("3. Найти элемент\n");**

**printf("4. Удалить элемент\n");**

**printf("5. Выход\n");**

**printf("Выберите операцию: ");**

**scanf("%d", &choice);**

**switch (choice) {**

**case 1:**

**spstore();**

**break;**

**case 2:**

**review();**

**break;**

**case 3:**

**printf("Введите название объекта для поиска: ");**

**scanf("%s", name);**

**found = find(name);**

**if (found != NULL) {**

**printf("Найденный элемент: Имя - %s, Приоритет - % d\n", found->inf, found->priority);**

**}**

**break;**

**case 4:**

**printf("Введите название объекта для удаления: ");**

**scanf("%s", name);**

**del(name);**

**break;**

**case 5:**

**exit(0);**

**default:**

**printf("Неверный выбор. Пожалуйста, выберите снова.\n");**

**}**

**}**

**return 0;**

**}**

**Задание 2**:

**#define \_CRT\_SECURE\_NO\_WARNINGS**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <string.h>**

**#include <locale.h>**

**struct node**

**{**

**char inf[256];**

**struct node\* next;**

**};**

**struct node\* front = NULL, \* rear = NULL;**

**void enqueue(void);**

**void dequeue(void);**

**void display(void);**

**struct node\* get\_struct(void)**

**{**

**struct node\* p = NULL;**

**char s[256];**

**if ((p = (struct node\*)malloc(sizeof(struct node))) == NULL)**

**{**

**printf("Ошибка при распределении памяти\n");**

**exit(1);**

**}**

**printf("Введите название объекта: \n");**

**scanf("%s", s);**

**if (\*s == 0)**

**{**

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

**return NULL;**

**}**

**strcpy(p->inf, s);**

**p->next = NULL;**

**return p;**

**}**

**void enqueue(void)**

**{**

**struct node\* p = NULL;**

**p = get\_struct();**

**if (rear == NULL && p != NULL)**

**{**

**front = p;**

**rear = p;**

**}**

**else if (rear != NULL && p != NULL)**

**{**

**rear->next = p;**

**rear = p;**

**}**

**return;**

**}**

**void dequeue(void)**

**{**

**if (front == NULL)**

**{**

**printf("Очередь пуста\n");**

**return;**

**}**

**struct node\* temp = front;**

**front = front->next;**

**free(temp);**

**if (front == NULL)**

**rear = NULL;**

**}**

**void display(void)**

**{**

**struct node\* struc = front;**

**if (front == NULL)**

**{**

**printf("Очередь пуста\n");**

**}**

**while (struc)**

**{**

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

**struc = struc->next;**

**}**

**return;**

**}**

**int main()**

**{**

**setlocale(LC\_ALL, "Rus");**

**int choice;**

**while (1)**

**{**

**printf("\nОперации над очередью:\n");**

**printf("1. Добавить элемент в очередь\n");**

**printf("2. Удалить элемент из очереди\n");**

**printf("3. Просмотреть содержимое очереди\n");**

**printf("4. Выход\n");**

**printf("Введите выбор: ");**

**scanf("%d", &choice);**

**switch (choice)**

**{**

**case 1:**

**enqueue();**

**break;**

**case 2:**

**dequeue();**

**break;**

**case 3:**

**display();**

**break;**

**case 4:**

**exit(0);**

**default:**

**printf("Неправильный выбор\n");**

**}**

**}**

**return 0;**

**}**

**Задание 3**:

**#define \_CRT\_SECURE\_NO\_WARNINGS**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <string.h>**

**#include <locale.h>**

**struct node**

**{**

**char inf[256];**

**struct node\* next;**

**};**

**struct node\* top = NULL;**

**void push(void);**

**void pop(void);**

**void display(void);**

**struct node\* get\_struct(void)**

**{**

**struct node\* p = NULL;**

**char s[256];**

**if ((p = (struct node\*)malloc(sizeof(struct node))) == NULL)**

**{**

**printf("Ошибка при распределении памяти\n");**

**exit(1);**

**}**

**printf("Введите название объекта: \n");**

**scanf("%s", s);**

**if (\*s == 0)**

**{**

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

**return NULL;**

**}**

**strcpy(p->inf, s);**

**p->next = NULL;**

**return p;**

**}**

**void push(void)**

**{**

**struct node\* p = NULL;**

**p = get\_struct();**

**if (p != NULL)**

**{**

**p->next = top;**

**top = p;**

**}**

**}**

**void pop(void)**

**{**

**if (top == NULL)**

**{**

**printf("Стек пуст\n");**

**return;**

**}**

**struct node\* temp = top;**

**top = top->next;**

**free(temp);**

**}**

**void display(void)**

**{**

**struct node\* struc = top;**

**if (top == NULL)**

**{**

**printf("Стек пуст\n");**

**}**

**while (struc)**

**{**

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

**struc = struc->next;**

**}**

**}**

**int main()**

**{**

**setlocale(LC\_ALL, "Rus");**

**int choice;**

**while (1)**

**{**

**printf("\nОперации над стеком:\n");**

**printf("1. Добавить элемент в стек\n");**

**printf("2. Удалить элемент из стека\n");**

**printf("3. Просмотреть содержимое стека\n");**

**printf("4. Выход\n");**

**printf("Введите выбор: ");**

**scanf("%d", &choice);**

**switch (choice)**

**{**

**case 1:**

**push();**

**break;**

**case 2:**

**pop();**

**break;**

**case 3:**

**display();**

**break;**

**case 4:**

**exit(0);**

**default:**

**printf("Неправильный выбор\n");**

**}**

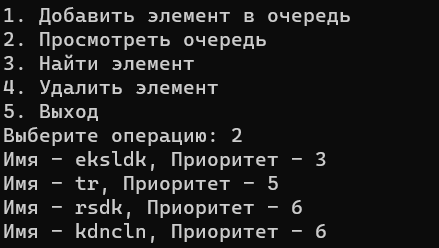
**}**

**return 0;**

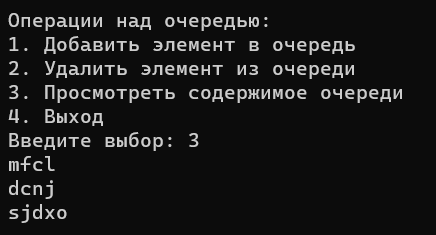
**}**

**Результаты работы программы**

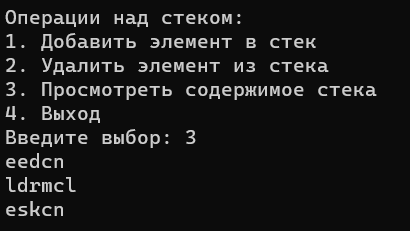
**Задание 1**:



**Задание 2**:



**Задание 3**:



**Вывод**

В ходе выполнения данной лабораторной работы были получены практические навыки реализации базовых динамических структур данных - очереди, стека и приоритетной очереди на основе односвязных списков на языке C.

Были изучены основные принципы организации односвязных списков, реализованы функции для работы с динамической памятью при создании и удалении элементов.