МИНИСТЕРВСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №3

по курсу «Логика и ОА в ИЗ»

на тему «Динамические списки»

Выполнили:

Студенты группы 23ВВВ2

Севостьянов Олег

Кандаков Андрей

Принял:

Митрохин М.А.

Юрова О.В.

Пенза 2024

**Цель работы**

Изучение динамических списков.

**Лабораторное задание**

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).

Задание 2\*:

На основе приведенного кода реализуйте структуру данных *Очередь*.

Задание 3\*:

На основе приведенного кода реализуйте структуру данных *Стек*.

**Пояснение к коду**

Задание 1:

**Структура данных node**

struct node {

char inf[256]; *// полезная информация*

int priority; *// приоритет*

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

};

* **inf** — массив символов длиной 256, который хранит "полезную информацию" или имя объекта.
* **priority** — целое число, представляющее приоритет текущего элемента. Этот параметр используется для сортировки элементов.
* **next** — указатель на следующий элемент в списке. Так как мы используем односвязный список, каждый элемент знает только о следующем.

**Объявление указателей head и last**

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

* **head** — указатель на первый элемент списка.
* **last** — указатель на последний элемент списка. Эти указатели помогают эффективно управлять началом и концом списка.

**Прототипы функций**

void spstore\_priority(void);

void review(void);

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

* **spstore\_priority** — функция для добавления нового элемента в список с учетом приоритета.
* **review** — функция для вывода всех элементов списка на экран.
* **get\_struct\_priority** — функция для создания и инициализации нового элемента с указанным приоритетом.

**Функция создания элемента с приоритетом get\_struct\_priority**

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

struct node\* p = NULL;

char s[256];

int priority;

* **p** — временный указатель для создания нового элемента.
* **s** — массив для хранения строки с именем объекта, введенной пользователем.
* **priority** — переменная для хранения приоритета объекта, также введенного пользователем.

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

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

exit(1);

}

* Выделяется память для нового элемента node. Если выделение памяти не удалось, программа завершится с сообщением об ошибке.

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

scanf("%s", s);

strcpy(p->inf, s);

* Пользователю предлагается ввести название объекта, которое сохраняется в s, а затем копируется в p->inf.

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

scanf("%d", &priority);

p->priority = priority;

* Пользователь вводит приоритет объекта, который сохраняется в priority и передается в p->priority.

p->next = NULL;

return p;

}

* **p->next** инициализируется как NULL, так как новый элемент вначале не указывает ни на какой следующий элемент.
* Возвращается указатель на созданный элемент p.

**Функция добавления элемента с учетом приоритета spstore\_priority**

void spstore\_priority(void) {

struct node\* p = NULL;

p = get\_struct\_priority();

* Создается новый элемент с помощью get\_struct\_priority() и сохраняется в p.

if (head == NULL) { *// Если список пуст, делаем новый элемент головой*

head = p;

last = p;

}

* Если список пуст (то есть head равен NULL), новый элемент становится началом и концом списка, и указатели head и last указывают на него.

else if (p->priority > head->priority) { *// Если элемент имеет больший приоритет, чем текущая голова*

p->next = head;

head = p;

}

* Если новый элемент имеет более высокий приоритет, чем текущий head, он становится новым началом списка. Указатель p->next указывает на старый head, и теперь head указывает на p.

else {

struct node\* current = head;

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

current = current->next;

}

p->next = current->next;

current->next = p;

if (p->next == NULL) {

last = p;

}

}

* В противном случае программа находит подходящую позицию для вставки нового элемента, сравнивая его приоритет с приоритетом последующих элементов. Когда позиция найдена, новый элемент p вставляется в список.
* Если p вставляется в конец списка, указатель last обновляется, чтобы указывать на p.

**Функция просмотра содержимого списка review**

void review(void) {

struct node\* struc = head;

if (head == NULL) {

printf("Список пуст\n");

return;

}

* Проверяется, пуст ли список (то есть равен ли head NULL). Если пуст, выводится сообщение "Список пуст".

while (struc) {

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

struc = struc->next;

}

return;

}

* В цикле while осуществляется проход по списку с выводом имени и приоритета каждого элемента. Цикл завершается, когда указатель struc становится NULL, что означает, что достигнут конец списка.

Задание 2:

Задание 3:

**Результат работы программы**

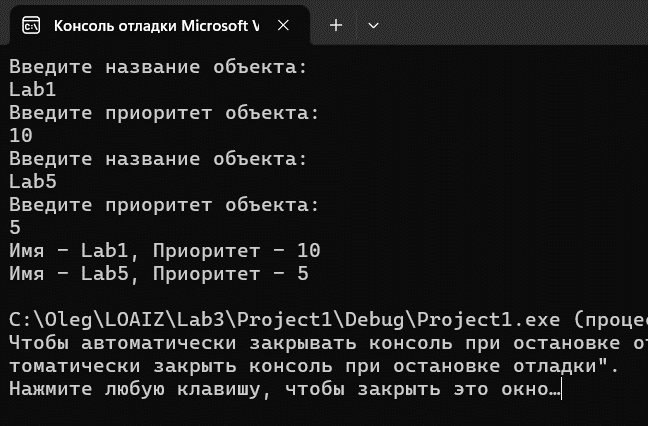


Рисунок 1 – результат работы программы (Задание 1)

**Вывод**

В ходе работы изучены динамические списки.