#define \_CRT\_SECURE\_NO\_WARNINGS // Отключение предупреждений о небезопасных функциях в Visual Studio (например, scanf, strcpy)

#include <stdio.h> // Подключение библиотеки для ввода/вывода данных (например, printf, scanf)

#include <stdlib.h> // Подключение библиотеки для работы с динамическим выделением памяти (например, malloc, exit)

#include <string.h> // Подключение библиотеки для работы со строками (например, strcpy)

#include <windows.h> // Подключение библиотеки для работы с Windows (например, для изменения кодировки консоли)

struct node { // Определение структуры данных для элемента списка

char inf[256]; // Поле для хранения информации (например, название объекта)

int priority; // Поле для хранения приоритета объекта

struct node\* next; // Указатель на следующий элемент списка

};

struct node\* head = NULL, \* last = NULL; // Указатели на первый и последний элементы списка. Изначально список пуст.

// Объявления функций

void spstore\_priority(void); // Функция добавления элемента в список с учётом приоритета

void review(void); // Функция для просмотра содержимого списка

struct node\* get\_struct\_priority(void); // Функция для создания нового элемента с приоритетом

// Функция для создания нового элемента с приоритетом

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

struct node\* p = NULL; // Указатель на новый элемент списка

char s[256]; // Массив для хранения строки с названием объекта

int priority; // Переменная для хранения приоритета

// Динамическое выделение памяти для нового элемента

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

printf("Ошибка при распределении памяти\n"); // Если память не выделена, выводим ошибку

exit(1); // Завершаем программу с ошибкой

}

// Запрос информации у пользователя

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

scanf("%s", s); // Считываем строку с названием объекта

strcpy(p->inf, s); // Копируем строку в поле инфо нового элемента

// Запрос приоритета объекта

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

scanf("%d", &priority); // Считываем приоритет

p->priority = priority; // Присваиваем приоритет элементу

p->next = NULL; // Устанавливаем указатель на следующий элемент как NULL (это последний элемент на данный момент)

return p; // Возвращаем указатель на созданный элемент

}

// Функция для добавления элемента в список с учётом его приоритета

void spstore\_priority(void) {

struct node\* p = NULL; // Указатель на новый элемент

p = get\_struct\_priority(); // Создаем новый элемент с помощью функции get\_struct\_priority

// Если список пуст, новый элемент становится первым

if (head == NULL) {

head = p; // Новый элемент становится головой списка

last = p; // Новый элемент также становится последним элементом

}

// Если новый элемент имеет больший приоритет, чем текущая голова

else if (p->priority > head->priority) {

p->next = head; // Новый элемент указывает на текущую голову

head = p; // Новый элемент становится головой списка

}

else {

// Ищем место для нового элемента по приоритету

struct node\* current = head; // Начинаем с головы списка

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

current = current->next; // Идем по списку, пока не найдем элемент с меньшим приоритетом

}

p->next = current->next; // Новый элемент указывает на следующий элемент после текущего

current->next = p; // Текущий элемент указывает на новый элемент

// Если новый элемент стал последним, обновляем указатель last

if (p->next == NULL) {

last = p;

}

}

return; // Функция завершена

}

// Функция для просмотра содержимого списка

void review(void) {

struct node\* struc = head; // Начинаем с головы списка

// Если список пуст, выводим сообщение

if (head == NULL) {

printf("Список пуст\n");

return; // Завершаем функцию

}

// Проходим по всем элементам списка и выводим информацию

while (struc) {

printf("Имя - %s, Приоритет - %d\n", struc->inf, struc->priority); // Выводим информацию о текущем элементе

struc = struc->next; // Переходим к следующему элементу

}

return; // Функция завершена

}

// Главная функция программы

int main() {

SetConsoleOutputCP(1251); // Устанавливаем кодировку вывода в консоли (для корректного отображения русских символов)

SetConsoleCP(1251); // Устанавливаем кодировку ввода в консоли (для корректного ввода русских символов)

spstore\_priority(); // Добавляем первый элемент в список с приоритетом

spstore\_priority(); // Добавляем второй элемент в список с приоритетом

review(); // Просматриваем и выводим содержимое списка

return 0; // Завершаем программу

}