Министерство науки и высшего образования Российской Федерации

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 группы 22ВВВ3:

Куракин Н.Н.

Майоров Н.А.

Матюшин К. М.

Приняли:

Юрова О.В.

Акифьев И.В.

Пенза 2023

**Название**

Динамические списки.

**Цель работы**

Научиться работать с динамическими списками, реализовать некоторые структуры данных.

**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).

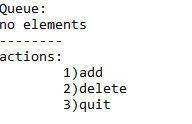
*Задание 2:*

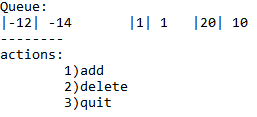
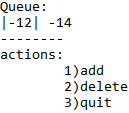
На основе приведенного кода реализуйте структуру данных Очередь.

*Задание 3:*

На основе приведенного кода реализуйте структуру данных Стек.

Задание 1:

Реализована приоритетная очередь. Программа имеет следующий интерфейс:  
  
После добавления элементов с приоритетами получается следующая очередь. (|P| N, где P – приоритет, N – число)

  
Удаление элемента из очереди происходит следующим образом: удаляется число с наибольшим приоритетом. После выполнения удаления очередь приобретает вид.  


Задание 2:

На основе приведённого в лабораторной работе кода реализована структура Очередь. Добавление элемента осуществляется в конец очереди. Удаляется элемент, находящийся в начала очереди.

Задание 3:

На основе приведённого в лабораторной работе кода реализована структура Стек. Добавление элемента осуществляется в конец стека. Удаляется элемент, находящийся в конце стека.

**Вывод:** в ходе выполнения лабораторной работы были получены/повторены навыки реализации структур данных Очередь, Стек. Была изучена такая разновидность структуры Очередь, как Приоритетная очередь.

**Листинг**:

Код 1:

#define QUEUE\_SIZE 100

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

struct QueueItem {

int priority, value;

};

void enterData(struct QueueItem\* queue, int currentID) {

int tempValue = 0, tempPriority = 0;

printf("\n--------\nValue: ");

scanf("%d", &tempValue);

printf("Priority of value: ");

scanf("%d", &tempPriority);

queue[currentID].priority = tempPriority;

queue[currentID].value = tempValue;

}

void showQueue(struct QueueItem\* queue, int currentID) {

system("cls");

printf("Queue:\n");

if (!currentID) printf("no elements");

else

for (int i = 0; i < currentID; i++) {

printf("|%d| %d\t", queue[i].priority, queue[i].value);

}

printf("\n--------");

}

void bubbleSort(struct QueueItem\* queue, int size) {

int gap = size;

int swapped = 1;

while (gap > 1 || swapped) {

gap = (gap \* 1000) / 1247; // Уменьшаем шаг на 1000/1247

if (gap < 1) {

gap = 1;

}

swapped = 0;

for (int i = 0; i < size - gap; i++) {

if (queue[i].priority > queue[i + gap].priority) {

struct QueueItem temp = queue[i];

queue[i] = queue[i + gap];

queue[i + gap] = temp;

swapped = 1;

}

}

}

}

int main() {

struct QueueItem queue[QUEUE\_SIZE] = { 0 };

int currentID = 0;

while (1) {

showQueue(queue, currentID);

printf("\nactions:\n\t1)add\n\t2)delete\n\t3)quit");

switch (\_getch()) {

case '1':

if (currentID > QUEUE\_SIZE) break;

enterData(queue, currentID++);

bubbleSort(queue, currentID);

showQueue(queue, currentID);

break;

case '2':

if (currentID) {

queue[currentID].value = 0;

queue[currentID--].priority = 0;

}

showQueue(queue, currentID);

break;

case '3':

showQueue(queue, currentID++);

return 0;

}

}

}

Код 2:

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <malloc.h>

#include <string.h>

#include <stdlib.h>

#include <locale.h>

#include <conio.h>

struct node

{

char inf[256]; // полезная информация

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

};

char find\_el[256];

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

int dlinna = 0;

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

{

struct node\* p = NULL;

char s[256];

if ((p = (struct 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 очередь элемента (в конец)\*/

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)

{

last->next = p;

last = p;

}

return;

}

/\* Просмотр содержимого очереди. \*/

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) // функция нахождения элемента

{

struct node\* struc = head;

if (head == NULL)

{

printf("Очередь пуста\n");

}

while (struc)

{

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

{

return struc;

}

struc = struc->next;

}

printf("Элемент не найден\n");

return NULL;

}

/\* Удаление элемента по содержимому. \*/

void del()

{

if (head == NULL) // если голова списка равна NULL, то список пуст

{

printf("Очередь пуста");

\_getch();

return;

}

struct node\* tmp = head->next;

free(head);

head = tmp;

}

int main() {

setlocale(LC\_ALL, "RU");

while (1) {

system("cls");

printf("действия:\n\t1)добавить\n\t2)посмотреть\n\t3)удалить");

switch (\_getch()) {

case '1':

system("cls");

spstore();

system("cls");

break;

case '2':

system("cls");

review();

printf("Нажать кнопку для продолжения");

\_getch();

system("cls");

break;

case '3':

system("cls");

del();

system("cls");

break;

}

}

}

Код 3:

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <malloc.h>

#include <string.h>

#include <stdlib.h>

#include <locale.h>

#include <conio.h>

struct node

{

char inf[256]; // полезная информация

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

};

char find\_el[256];

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

int dlinna = 0;

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

{

struct node\* p = NULL;

char s[256];

if ((p = (struct 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 очередь элемента (в конец)\*/

void spstore(void)

{

struct node\* p = NULL;

p = get\_struct();

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

{

head = p;

prelast = p;

last = p;

}

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

{

last->next = p;

prelast = last;

last = p;

last->next = NULL;

}

dlinna++;

return;

}

/\* Просмотр содержимого очереди. \*/

void review(void)

{

struct node\* struc = head;

if (head == NULL)

{

printf("Стек пуст\n");

}

while (struc)

{

printf("Имя - %s, \n", struc->inf);

struc = struc->next;

}

return;

}

/\* Поиск элемента по содержимому. \*/

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

{

struct node\* struc = head;

if (head == NULL)

{

printf("Стек пуст\n");

}

while (struc)

{

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

{

return struc;

}

struc = struc->next;

}

printf("Элемент не найден\n");

return NULL;

}

/\* Удаление элемента по содержимому. \*/

void del()

{

if (head == NULL) // если голова списка равна NULL, то список пуст

{

printf("Стек пуст");

\_getch();

return;

}

free(last);

if (dlinna-- == 1) {

head = NULL;

last = NULL;

prelast = NULL;

return;

}

prelast->next = NULL;

prelast = head;

for (int i = 1; i < dlinna-1;i++) {

prelast = prelast->next;

};

last = dlinna != 1 ? prelast->next : head;

}

int main() {

setlocale(LC\_ALL, "RU");

while (1) {

system("cls");

printf("действия:\n\t1)добавить\n\t2)посмотреть\n\t3)удалить");

switch (\_getch()) {

case '1':

system("cls");

spstore();

system("cls");

break;

case '2':

system("cls");

review();

printf("Нажать кнопку для продолжения");

\_getch();

system("cls");

break;

case '3':

system("cls");

del();

system("cls");

break;

}

}

}