Министерство образования Российской Федерации

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 21вв1.1:**

Кузнецов А.С.

Гришин С.В.

**Приняли**

Юрова О.В.

Акифьев И.В.

Пенза 2022

**Методические указания.**

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*.

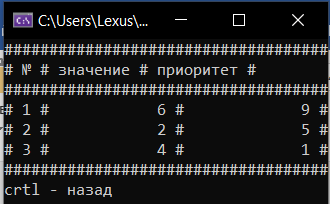
**Описание метода решения задачи:**

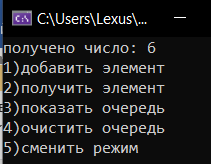
1)Создать односвязный список.

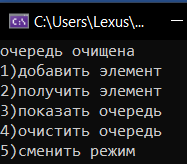
2) На его основе создать три типа очереди .

**Результаты работы программы:**

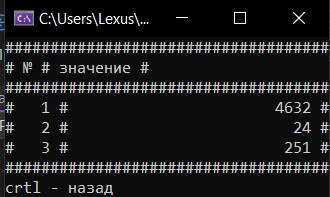
1)Реализована приоритетная очередь , путём добавления элемента в список в соответствии с приоритетом объекта

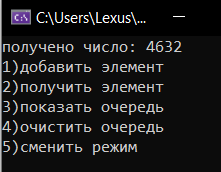


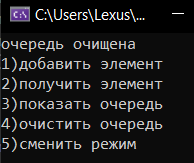




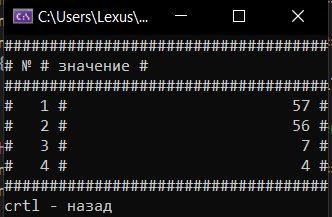
2) На основе приведенного кода реализуйте структуру данных очередь

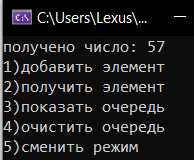


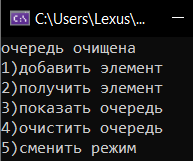




3) На основе приведенного кода реализуйте структуру данных стек







**Вывод:** в ходе выполнения лабораторной работы, мы научились работать с очередями. Результаты работы программ совпали с результатами трассировки, следовательно программы работают без ошибок.

**Листинг:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <stdint.h>

#include <malloc.h>

#include <Windows.h>

void hidecursore()

{

HANDLE consolehandle = GetStdHandle(STD\_OUTPUT\_HANDLE);

CONSOLE\_CURSOR\_INFO info;

info.dwSize = 100;

info.bVisible = FALSE;

SetConsoleCursorInfo(consolehandle, &info);

}

//опр н

typedef struct squep

{

struct squep\* next;

int zn;

int pr;

};

struct squep\* initp(int zn, int pr)

{

struct squep\* que;

que = (struct squep\*)malloc(sizeof(struct squep));

que->zn = zn;

que->pr = pr;

que->next = NULL;

return(que);

}

struct squep\* addst(squep\* qs, int zn, int pr)

{

struct squep\* que;

que = (struct squep\*)malloc(sizeof(struct squep));

que->zn = zn;

que->pr = pr;

que->next = qs;

return(que);

}

void addel(squep\* que, int zn, int pr)

{

squep\* temp = 0;

temp = que;

if (que->next == NULL)

{

squep\* tp = (struct squep\*)malloc(sizeof(squep));

tp->next = NULL;

tp->zn = zn;

tp->pr = pr;

que->next = tp;

}

else

{

while ((que->next != NULL) and (que->pr >= pr))

{

temp = que;

que = que->next;

}

if ((que->next == NULL) and (temp->pr != pr))

{

squep\* tp = (struct squep\*)malloc(sizeof(squep));

tp->next = NULL;

tp->zn = zn;

tp->pr = pr;

que->next = tp;

}

else

{

squep\* tp = (struct squep\*)malloc(sizeof(squep));

tp->next = que;

temp->next = tp;

tp->pr = pr;

tp->zn = zn;

}

}

}

void listprintp(squep\* que)

{

struct squep\* p;

p = que;

int i = 1;

printf("####################################\n");

printf("# № # значение # приоритет #\n");

printf("####################################\n");

do {

printf("#%2d # %12d # %13d #\n", i, p->zn, p->pr); // вывод значения элемента p

i++;

p = p->next; // переход к следующему узлу

} while (p != NULL);

printf("####################################");

printf("crtl - назад");

}

struct squep\* getp(squep\* que)

{

squep\* temp = que->next;

free(que);

return(temp);

}

//опр к

//очередь н

typedef struct sque

{

struct sque\* next;

int val;

};

struct sque\* get(sque\* que)

{

sque\* temp = que->next;

free(que);

return(temp);

}

void listprint(sque\* que)

{

struct sque\* p;

p = que;

int i = 1;

printf("####################################\n");

printf("# № # значение #\n");

printf("####################################\n");

do {

printf("#%4d # %26d #\n", i, p->val); // вывод значения элемента p

i++;

p = p->next; // переход к следующему узлу

} while (p != NULL);

printf("####################################");

printf("crtl - назад");

}

void addelemend(sque\* que, int number)

{

while (que->next != NULL)

{

que = que->next;

}

struct sque\* p;

sque\* temp = (struct sque\*)malloc(sizeof(sque));

temp->val = number;

que->next = temp;

temp->next = NULL;

}

struct sque\* init(int a) // а- значение первого узла

{

struct sque\* que;// выделение памяти под корень списка

que = (struct sque\*)malloc(sizeof(struct sque));

que->val = a;

que->next = NULL; // это последний узел списка

return(que);

}

//очередь к

//стек н

struct sque\* addelemendst(sque\* qs, int zn)

{

struct sque\* que;

que = (struct sque\*)malloc(sizeof(struct sque));

que->val = zn;

que->next = qs;

return(que);

}

//стек к

int main()

{

hidecursore();

system("mode con cols=36 lines=30");

system("chcp 1251");

system("cls");

struct squep\* ls = NULL;

struct sque\* l = NULL;

struct sque\* st = NULL;

bool cic = true;

int mod, colo = 0, colp = 0, cols = 0, zn = 0, pr = 0, maxpr = 0, res = 0;

l1:

printf("Выберите режим:\n1)очередь\n2)очередь с приоритетом\n3)стек\n4)выход\n");

scanf\_s("%d", &mod);

if ((mod < 1) or (mod > 4))

{

system("cls");

goto l1;

}

//очередь\/

if (mod == 1)

{

system("cls");

l3:

printf("1)добавить элемент\n2)получить элемент\n3)показать очередь\n4)очистить очередь\n5)сменить режим\n");

scanf\_s("%d", &mod);

if ((mod < 1) or (mod > 5))

{

system("cls");

goto l2;

}

if (mod == 1)

{

system("cls");

printf("Введите значение: ");

scanf\_s("%d", &zn);

if (colo == 0)

{

l = init(zn);

colo++;

}

else

{

addelemend(l, zn);

}

system("cls");

goto l3;

}

if (mod == 2)

{

if (l == NULL)

{

system("cls");

printf("Очередь пуста!\n");

goto l3;

}

else

{

system("cls");

res = l->val;

l = get(l);

colo--;

printf("получено чиcло: %d\n", res);

goto l3;

}

}

if (mod == 3)

{

system("cls");

if (l == NULL) { printf("Очередь пуста!\n"); goto l3; }

listprint(l);

while (cic == true) { Sleep(10); if (GetKeyState(17) < 0) { cic = false; } }

system("cls");

cic = true;

goto l3;

}

if (mod == 4)

{

system("cls");

if (l == NULL) { printf("Очередь пуста!\n"); goto l3; }

free(l);

l = NULL;

cols = 0;

printf("очередь очищена\n");

goto l3;

}

if (mod == 5) { system("cls"); goto l1; }

}

//очередь с приорететом\/

if (mod == 2)

{

system("cls");

l2:

printf("1)добавить элемент\n2)получить элемент\n3)показать очередь\n4)очистить очередь\n5)сменить режим\n");

scanf\_s("%d", &mod);

if ((mod < 1) or (mod > 5))

{

system("cls");

goto l2;

}

if (mod == 1)

{

system("cls");

printf("введите значение элемента:\n");

scanf\_s("%d", &zn);

printf("введите приоритет элемента\n(чем выше введённое значение,\n тем выше приоритет):\n");

scanf\_s("%d", &pr);

system("cls");

//ввод первого элемента\/

if (colp == 0)

{

ls = initp(zn, pr);

colp++;

maxpr = pr;

goto l2;

}

else

{

//ввод самого высокого приоритета\/

if (pr > maxpr)

{

ls = addst(ls, zn, pr);

colp++;

if (pr > maxpr) { maxpr = pr; }

goto l2;

}

else

{

addel(ls, zn, pr);

colp++;

if (pr > maxpr) { maxpr = pr; }

goto l2;

}

}

}

if (mod == 2)

{

if (ls == NULL)

{

system("cls");

printf("Очередь пуста!\n");

goto l2;

}

else

{

res = ls->zn;

ls = getp(ls);

colp--;

system("cls");

printf("получено чиcло: %d\n", res);

goto l2;

}

}

if (mod == 3)

{

if (ls == NULL)

{

system("cls");

printf("Очередь пуста!\n");

goto l2;

}

else

{

system("cls");

listprintp(ls);

while (cic == true) { Sleep(10); if (GetKeyState(17) < 0) { cic = false; } }

cic = true;

system("cls");

goto l2;

}

}

if (mod == 4)

{

system("cls");

if (ls == NULL) { printf("Очередь пуста!\n"); goto l2; }

free(ls);

ls = NULL;

colp = 0;

printf("очередь очищена\n");

goto l2;

}

if (mod == 5) { system("cls"); goto l1; }

}

//стек\/

if (mod == 3)

{

system("cls");

l4:

printf("1)добавить элемент\n2)получить элемент\n3)показать очередь\n4)очистить очередь\n5)сменить режим\n");

scanf\_s("%d", &mod);

if ((mod < 1) or (mod > 5))

{

system("cls");

goto l4;

}

if (mod == 1)

{

system("cls");

printf("Введите значение: ");

scanf\_s("%d", &zn);

if (cols == 0)

{

st = init(zn);

cols++;

}

else

{

st = addelemendst(st, zn);

cols++;

}

system("cls");

goto l4;

}

if (mod == 2)

{

if (st == NULL)

{

system("cls");

printf("Очередь пуста!\n");

goto l4;

}

else

{

system("cls");

res = st->val;

st = get(st);

cols--;

printf("получено чиcло: %d\n", res);

goto l4;

}

}

if (mod == 3)

{

system("cls");

if (st == NULL) { printf("Очередь пуста!\n"); goto l4; }

listprint(st);

while (cic == true) { Sleep(10); if (GetKeyState(17) < 0) { cic = false; } }

system("cls");

cic = true;

goto l4;

}

if (mod == 4)

{

system("cls");

if (st == NULL) { printf("Очередь пуста!\n"); goto l4; }

free(st);

st = NULL;

cols = 0;

printf("очередь очищена\n");

goto l4;

}

if (mod == 5) {

system("cls");

goto l1;

}

}

if (mod == 4) { exit; }

}