Пензенский государственный университет

Кафедра «Вычислительная техника»

ОТЧЕТ

по лабораторной работе № 3

по дисциплине: "Логика и ОА в ИЗ"

на тему: "Динамические списки"

Выполнили:

Кирилин М. С.

Николаев А. А.

Приняли:

Акифьев И.В.

Юрова О.В.

Пенза 2023

**Общие сведения.**

**Цель:** Научиться создавать Приоритетную Очередь. Научиться создавать структуру данных – Стек. Научиться создавать структуру данных – Очередь.

Список представляет собой последовательность элементов определенного типа. Простейший тип списка – линейный, когда для каждого из элементов, кроме последнего, имеется следующий, и для каждого, кроме первого имеется предыдущий элемент.

Возможна реализация списков посредством массивов или динамическая реализация.

Динамические списки относятся к динамическим структурам и используются, когда размер данных заранее неизвестен. Созданием динамических данных должна заниматься сама программа во время своего исполнения, этим достигается эффективное распределение памяти, но снижается эффективность доступа к элементам.

Динамические структуры данных отличаются от статических двумя основными свойствами:

1) в них нельзя обеспечить хранение в заголовке всей информации о структуре, поэтому каждый элемент должен содержать информацию, логически связывающую его с другими элементами структуры;

2) для них зачастую не удобно использовать единый массив смежных элементов памяти, поэтому необходимо предусматривать ту или иную схему динамического управления памятью.

Для обращения к динамическим данным применяют указатели.

Набор операций над списком будет включать добавление и удаление элементов, поиск элементов списка.

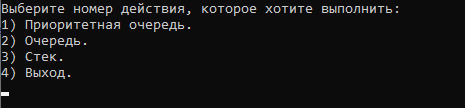
Различают односвязные, двусвязные и циклические списки.

В простейшем случае каждый элемент содержит всего одну ссылку на следующий элемент, такой список называется односвязным.

В простейшем случае для создания элемента списка используется структура, в которой объединяются полезная информация и ссылка на следующий элемент списка:

**Задание**

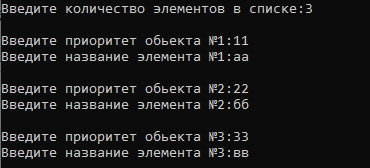
1. Реализовать приоритетную очередь, путём добавления элемента в список в соответствии с приоритетом объекта (т.е. объект с большим приоритетом становится перед объектом с меньшим приоритетом).



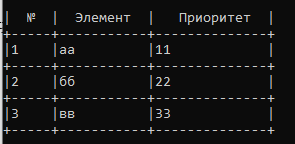
На фото представлено меню программы.

В данной программе реализовано: [1] Приоритетная очередь; [2] Очередь; [3] Стек; [4] Выход.

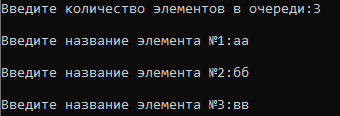
Попробуем добавить пару элементов:



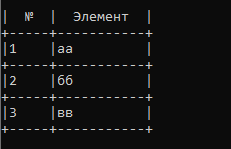
Как можно увидеть. Программа выводит числа, в зависимости от их приоритета.



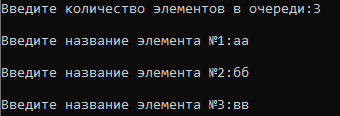
Попробуем [2]:



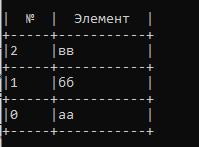
Как можно увидеть. Программа выводит числа, в порядке очереди.



Попробуем [3]:



Как можно увидеть. Программа выводит числа в стек.



**Вывод:**

Научились создавать Приоритетную Очередь. Научились создавать структуру данных – Стек. Научились создавать структуру данных – Очередь.

**Листинг**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

#include <Windows.h>

struct node

{

char inf[256]; // полезная информация

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

int prir;

};

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

int dlinna = 0;

struct node\* get\_struct(int pr)

{

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);

}

scanf(" %s", &s);

if (\*s == 0)

{

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

return NULL;

}

strcpy(p->inf, s);

p->next = NULL;

p->prir = pr;

return p; // возвращаем указатель на созданный элемент

}

void queue(struct node\*\* head, struct node\*\* last)

{

struct node\* temp = get\_struct(0);

if (\*head == NULL && temp != NULL)

{

\*head = temp;

\*last = temp;

}

else if (\*head != NULL && temp != NULL)

{

(\*last)->next = temp;

\*last = temp;

}

}

void ueue(struct node\*\* head)

{

struct node\* temp = get\_struct(0);

if (temp != NULL)

{

temp->next = \*head;

\*head = temp;

}

}

void prioritet(struct node\*\* head, int pr)

{

struct node\* prom = \*head;

struct node\* temp = get\_struct(pr);

if (\*head == NULL || (\*head)->prir >= pr)

{

temp->next = \*head;

\*head = temp;

}

else

{

while (prom->next != NULL && prom->next->prir < pr)

{

prom = prom->next;

}

temp->next = prom->next;

prom->next = temp;

}

}

void free\_queue(struct node\* head)

{

struct node\* current = head;

struct node\* next;

while (current != NULL)

{

next = current->next;

free(current);

current = next;

}

}

void main()

{

int f = 0;

int i = 0;

SetConsoleCP(1251);//Установка языка

SetConsoleOutputCP(1251);

while (f != 4)

{

system("cls");

printf("Выберите номер действия, которое хотите выполнить:\n");

printf("1) Приоритетная очередь.\n");

printf("2) Очередь.\n");

printf("3) Стек.\n");

printf("4) Выход.\n");

scanf("%d", &f);

switch (f)

{

case 1:

{

system("cls");

int pr = NULL;

printf("Введите количество элементов в списке:");

scanf("%d", &dlinna);

for (int i = 0; i < dlinna; i++)

{

printf("\nВведите приоритет обьекта №%d:", i + 1);

scanf("%d", &pr);

if (pr < 0)

{

printf(" Ошибка при записи приоритета\n");

\_getch();

return;

}

printf("Введите название элемента №%d:", i + 1);

prioritet(&head, pr);

}

printf("\n| № | Элемент | Приоритет |\n");

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

struct node\* current = head;

while (current != NULL)

{

i++;

printf("|%-5d|%-11s|%-14d|\n", i, current->inf, current->prir);

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

current = current->next;

}

\_getch();

return;

}

case 2:

{

system("cls");

printf("Введите количество элементов в очереди:");

scanf("%d", &dlinna);

for (int i = 0; i < dlinna; i++)

{

printf("\nВведите название элемента №%d:", i + 1);

queue(&head, &last);

}

printf("\n| № | Элемент |\n");

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

struct node\* urrent = head;

while (urrent != NULL)

{

i++;

printf("|%-5d|%-11s|\n", i, urrent->inf);

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

urrent = urrent->next;

}

\_getch();

return;

}

case 3:

{

system("cls");

printf("Введите количество элементов в очереди:");

scanf("%d", &dlinna);

for (int i = 0; i < dlinna; i++)

{

printf("\nВведите название элемента №%d:", i + 1);

ueue(&head);

}

printf("\n| № | Элемент |\n");

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

struct node\* rrent = head;

while (rrent != NULL)

{

dlinna--;

printf("|%-5d|%-11s|\n", dlinna, rrent->inf);

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

rrent = rrent->next;

}

\_getch();

return;

}

case 4:

{

system("cls");

printf("ВЫХОД\n");

return;

default:

printf("Неверный выбор\n");

return;

}

}

}

}