Министерство образования Российской федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЁТ**

по лабораторной работе №3

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Динамические списки»

Выполнили ст. группы 22ВВВ1:

Лёвин А.Д.

Колобов И.О.

Приняли:

К.э.н., доцент Акифьев И. В.

К.т.н., доцент Юрова О. В.

Пенза 2023

**Цель работы:**

Цель данной лабораторной работы заключается в изучении и практическом применении динамических списков в программировании .

**Лабораторное 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).

2. \* На основе приведенного кода реализуйте структуру данных Очередь.

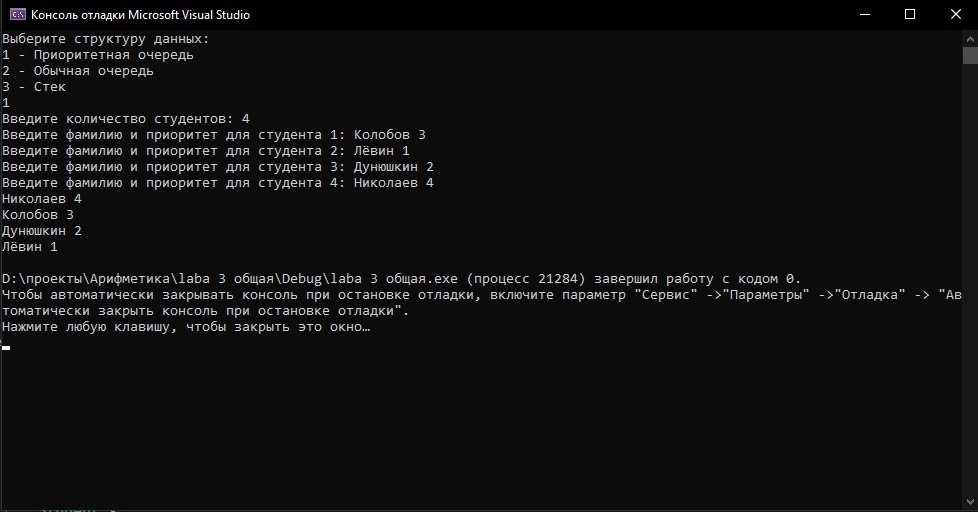
3. \* На основе приведенного кода реализуйте структуру данных Стек.

**Ход работы:**

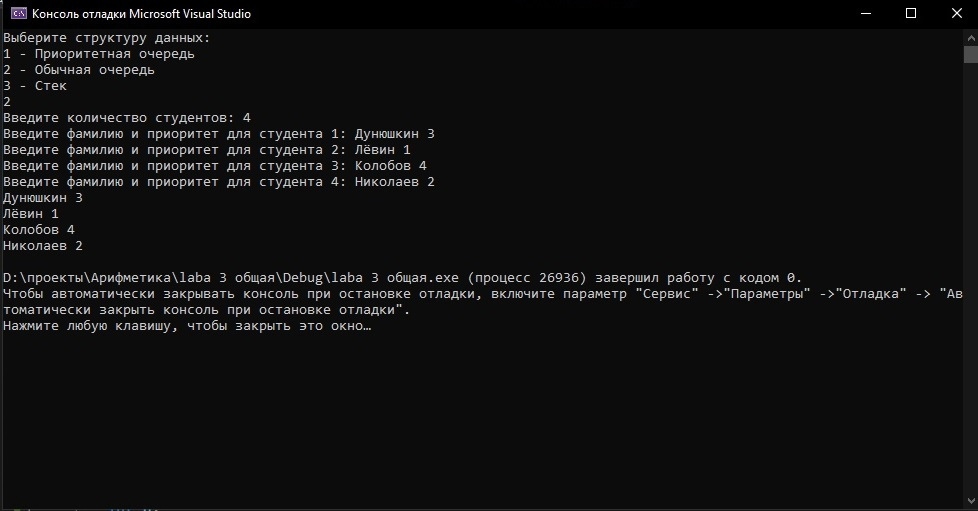
Инициализируем структуры данных Student, Queue и Stack для реализации приоритетной очереди, очереди и стека соответственно.  
Далее реализуем функции для работы с ними (сравнение для приоритетной очереди, добавление и извлечение эл-та для очереди и стека)  
Дальше в функции main мы выбираем какую структуру данных будем использовать.  
Приоритетная очередь:  
-Считывается количество студентов  
-Выделяется память для массива студентов  
-В цикле считываются данные о студентах  
-Вызывается сортировка qsort по приоритету  
-Выводятся отсортированные студенты  
Обычная очередь:  
-Создается очередь заданного размера  
-В цикле считываются студенты и добавляются в -очередь через enqueue  
-В цикле извлекаются студенты из очереди через dequeue и выводятся  
Стек:  
-Создается стек заданного размера  
-В цикле считываются студенты и добавляются в -стек через push  
-В цикле студенты извлекаются из стека через pop и выводятся  
Если выбрана неверная структура данных - выводится сообщение об ошибке.

**Результаты работы программы:**

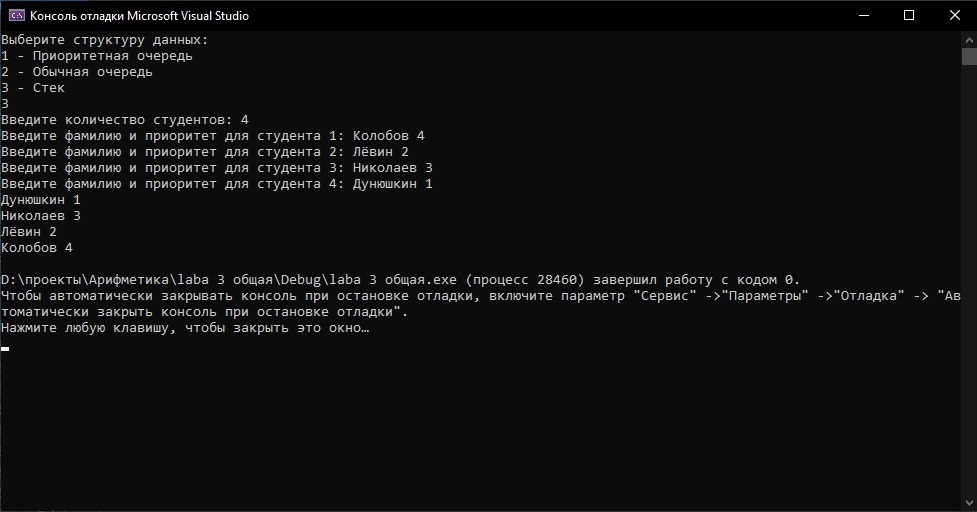
**Приоритетная очередь :**



**Очередь:**



**Стек:**



**Вывод:**

В ходе лабораторной работы научились создавать и работать с приоритетной очередью, структурой данных – Стек и структурой данных – Очередь.

**Листинг:**

[#define](https://vk.com/im?sel=268920549&st=%23define) \_CRT\_SECURE\_NO\_WARNINGS  
  
[#include](https://vk.com/im?sel=268920549&st=%23include) <stdio.h>  
[#include](https://vk.com/im?sel=268920549&st=%23include) <stdlib.h>  
[#include](https://vk.com/im?sel=268920549&st=%23include) <string.h>  
[#include](https://vk.com/im?sel=268920549&st=%23include) <windows.h>  
  
// Структуры данных  
typedef struct {  
char lastName[100];  
int priority;  
} Student;  
  
typedef struct {  
Student\* data;  
int front, rear, size, capacity;  
} Queue;  
  
typedef struct {  
Student\* data;  
int top, capacity;  
} Stack;  
  
int compareStudents(const void\* a, const void\* b) {  
const Student\* studentA = (const Student\*)a;  
const Student\* studentB = (const Student\*)b;  
  
if (studentA->priority > studentB->priority)  
return -1;  
else if (studentA->priority < studentB->priority)  
return 1;  
else  
return 0;  
}  
  
// Функции для работы с очередью  
//Создание очереди  
Queue\* createQueue(int capacity) {  
Queue\* q = (Queue\*)malloc(sizeof(Queue));  
q->data = (Student\*)malloc(capacity \* sizeof(Student));  
q->front = 0;  
q->rear = -1;  
q->size = 0;  
q->capacity = capacity;  
return q;  
}  
  
//Добавление эл-та в очередь  
void enqueue(Queue\* q, Student s) {  
if (q->size == q->capacity) {  
printf("Очередь заполнена!");  
return;  
}  
  
q->rear = (q->rear + 1) % q->capacity;  
q->data[q->rear] = s;  
q->size++;  
}  
  
//Извлечение эл-та в очередь  
Student dequeue(Queue\* q) {  
Student s;  
if (q->size == 0) {  
printf("Очередь пуста!");  
}  
else {  
s = q->data[q->front];  
q->front = (q->front + 1) % q->capacity;  
q->size--;  
}  
  
return s;  
}  
  
// Функции для работы со стеком  
// Создание стека  
Stack\* createStack(int capacity) {  
Stack\* s = (Stack\*)malloc(sizeof(Stack));  
s->data = (Student\*)malloc(capacity \* sizeof(Student));  
s->top = -1;  
s->capacity = capacity;  
return s;  
}  
  
// Добавление эл-та в стек  
void push(Stack\* s, Student student) {  
if (s->top == s->capacity - 1) {  
printf("Стек переполнен!\n");  
return;  
}  
  
s->top++;  
s->data[s->top] = student;  
}  
  
// Извлечение эл-та из стека  
Student pop(Stack\* s) {  
if (s->top == -1) {  
printf("Стек пуст!\n");  
Student temp = { "", 0 }; // инициализируем пустым студентом  
return temp;  
}  
  
Student temp = s->data[s->top];  
s->top--;  
return temp;  
}  
  
int main() {  
SetConsoleCP(1251);//Установка языка  
SetConsoleOutputCP(1251);  
int choice;  
  
printf("Выберите структуру данных:\n");  
printf("1 - Приоритетная очередь\n");  
printf("2 - Обычная очередь\n");  
printf("3 - Стек\n");  
  
scanf("%d", &choice);  
  
if (choice == 1) {  
int numStudents;  
printf("Введите количество студентов: ");  
scanf("%d", &numStudents);  
  
Student\* students = (Student\*)malloc(numStudents \* sizeof(Student));  
  
for (int i = 0; i < numStudents; i++) {  
printf("Введите фамилию и приоритет для студента %d: ", i + 1);  
scanf("%s %d", students[i].lastName, &students[i].priority);  
}  
  
qsort(students, numStudents, sizeof(Student), compareStudents);  
  
for (int i = 0; i < numStudents; i++) {  
printf("%s %d\n", students[i].lastName, students[i].priority);  
}  
  
free(students);  
  
return 0;  
}  
else if (choice == 2) {  
Queue\* q = createQueue(100);  
  
int n;  
printf("Введите количество студентов: ");  
scanf("%d", &n);  
  
for (int i = 0; i < n; i++) {  
Student s;  
printf("Введите фамилию и приоритет для студента %d: ", i + 1);  
scanf("%s %d", s.lastName, &s.priority);  
  
enqueue(q, s);  
}  
  
while (q->size > 0) {  
Student s = dequeue(q);  
printf("%s %d\n", s.lastName, s.priority);  
}  
  
return 0;  
  
}  
else if (choice == 3) {  
Stack\* s = createStack(100);  
  
int n;  
printf("Введите количество студентов: ");  
scanf("%d", &n);  
  
for (int i = 0; i < n; i++) {  
Student student;  
printf("Введите фамилию и приоритет для студента %d: ", i + 1);  
scanf("%s %d", &student.lastName, &student.priority);  
  
push(s, student);  
}  
  
while (s->top != -1) {  
Student temp = pop(s);  
printf("%s %d\n", temp.lastName, temp.priority);  
}  
  
return 0;  
  
}  
else {  
printf("Неверный выбор!");  
}  
  
return 0;  
}