Министерство науки и высшего 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:

Уткин М.М.

Саветкин Д.Д.

Соколовский Е.В

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.

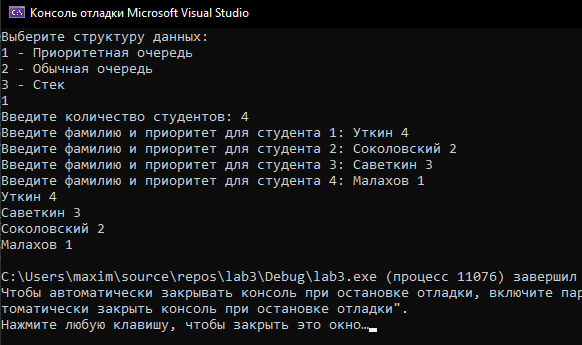
3. \* На основе приведенного кода реализуйте структуру данных Стек.

**Ход работы:**

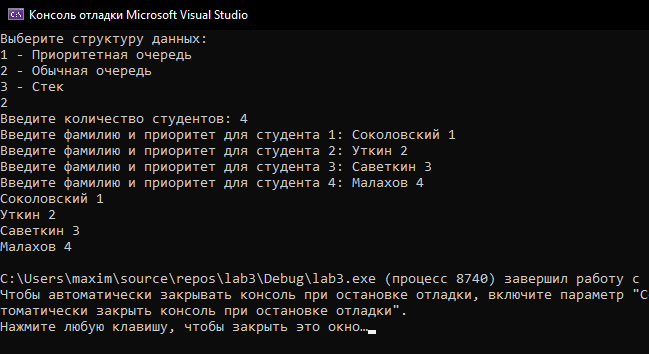
Инициализируем структуры данных Student, Queue и Stack для реализации приоритетной очереди, очереди и стека соответственно.  
Далее реализуем функции для работы с ними (сравнение для приоритетной очереди, добавление и извлечение эл-та для очереди и стека)  
Дальше в функции main мы выбираем какую структуру 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:**

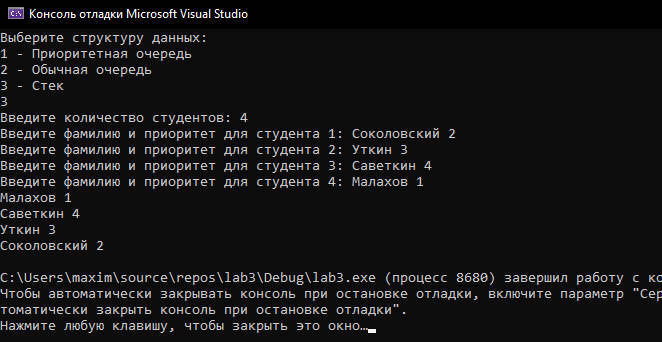
**Приоритетная очередь :**



**Очередь:**



**Стек:**



**Вывод:**

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

**Листинг:**

#define\_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<windows.h>

typedefstruct {

char lastName[100];

int priority;

} Student;

typedefstruct {

Student\* data;

int front, rear, size, capacity;

} Queue;

typedefstruct {

Student\* data;

int top, capacity;

} Stack;

int compareStudents(constvoid\* a, constvoid\* b) {

constStudent\* studentA = (constStudent\*)a;

constStudent\* studentB = (constStudent\*)b;

return studentB->priority - studentA->priority;

}

Queue\* createQueue(intcapacity) {

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;

}

void enqueue(Queue\* q, Students) {

if (q->size == q->capacity) {

printf("Очередьзаполнена!");

return;

}

q->rear = (q->rear + 1) % q->capacity;

q->data[q->rear] =s;

q->size++;

}

Student dequeue(Queue\* q) {

Student s;

if (q->size == 0) {

printf("Очередьпуста!");

}

else {

s =q->data[q->front];

q->front = (q->front + 1) % q->capacity;

q->size--;

}

return s;

}

Stack\* createStack(intcapacity) {

Stack\* s = (Stack\*)malloc(sizeof(Stack));

s->data = (Student\*)malloc(capacity \* sizeof(Student));

s->top = -1;

s->capacity = capacity;

return s;

}

void push(Stack\* s, Studentstudent) {

if (s->top == s->capacity - 1) {

printf("Стекпереполнен!\n");

return;

}

s->top++;

s->data[s->top] =student;

}

Student pop(Stack\* s) {

if (s->top == -1) {

printf("Стекпуст!\n");

Student temp = { "", 0 };

return temp;

}

Student temp = s->data[s->top];

s->top--;

return temp;

}

void updateStudent(Student\* student, constchar\* lastName, intpriority) {

strcpy(student->lastName, lastName);

student->priority = priority;

}

int main() {

SetConsoleCP(1251);

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 || choice > 3) {

printf("Неверныйвыбор!\n");

return 1;

}

int numStudents;

printf("Введитеколичествостудентов: ");

scanf("%d", &numStudents);

if (numStudents<= 0) {

printf("Количество студентов должно быть положительным числом!\n");

return 1;

}

Queue\* q = NULL;

Stack\* s = NULL;

if (choice == 1) {

Student\* students = (Student\*)malloc(numStudents \* sizeof(Student));

q = createQueue(100);

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);

enqueue(q, students[i]);

}

while (1) {

printf("Выберите действие:\n");

printf("1 - Добавить студента\n");

printf("2 - Удалить студента\n");

printf("3 - Вывести студентов\n");

printf("4 - Выход\n");

scanf("%d", &choice);

if (choice == 1) {

Studentstudent;

printf("Введите фамилию и приоритет для нового студента: ");

scanf("%s %d", student.lastName, &student.priority);

enqueue(q, student);

}

elseif (choice == 2) {

if (q->size > 0) {

Student removedStudent = dequeue(q);

printf("Удаленстудент: %s %d\n", removedStudent.lastName, removedStudent.priority);

}

else {

printf("Очередь пуста!\n");

}

}

elseif (choice == 3) {

printf("Список студентов в приоритетной очереди:\n");

for (int i = q->front, count = 0; count < q->size; i = (i + 1) % q->capacity, count++) {

printf("%s %d\n", q->data[i].lastName, q->data[i].priority);

}

}

elseif (choice == 4) {

break;

}

else {

printf("Неверныйвыбор!\n");

}

}

free(students);

free(q->data);

free(q);

}

elseif (choice == 2) {

q = createQueue(100);

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

Studentstudent;

printf("Введите фамилию и приоритет для студента %d: ", i + 1);

scanf("%s %d", student.lastName, &student.priority);

enqueue(q, student);

}

while (1) {

printf("Выберитедействие:\n");

printf("1 - Добавить студента\n");

printf("2 - Удалить студента\n");

printf("3 - Вывести студентов\n");

printf("4 - Выход\n");

scanf("%d", &choice);

if (choice == 1) {

Studentstudent;

printf("Введите фамилию и приоритет для нового студента: ");

scanf("%s %d", student.lastName, &student.priority);

enqueue(q, student);

}

elseif (choice == 2) {

if (q->size > 0) {

Student removedStudent = dequeue(q);

printf("Удаленстудент: %s %d\n", removedStudent.lastName, removedStudent.priority);

}

else {

printf("Очередь пуста!\n");

}

}

elseif (choice == 3) {

printf("Список студентов в обычной очереди:\n");

for (int i = q->front, count = 0; count < q->size; i = (i + 1) % q->capacity, count++) {

printf("%s %d\n", q->data[i].lastName, q->data[i].priority);

}

}

elseif (choice == 4) {

break;

}

else {

printf("Неверныйвыбор!\n");

}

}

free(q->data);

free(q);

}

elseif (choice == 3) {

s = createStack(100);

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

Studentstudent;

printf("Введите фамилию и приоритет для студента %d: ", i + 1);

scanf("%s %d", student.lastName, &student.priority);

push(s, student);

}

while (1) {

printf("Выберитедействие:\n");

printf("1 - Добавить студента\n");

printf("2 - Удалить студента\n");

printf("3 - Вывести студентов\n");

printf("4 - Выход\n");

scanf("%d", &choice);

if (choice == 1) {

Studentstudent;

printf("Введите фамилию и приоритет для нового студента: ");

scanf("%s %d", student.lastName, &student.priority);

push(s, student);

}

elseif (choice == 2) {

if (s->top != -1) {

Student removedStudent = pop(s);

printf("Удаленстудент: %s %d\n", removedStudent.lastName, removedStudent.priority);

}

else {

printf("Стекпуст!\n");

}

}

elseif (choice == 3) {

printf("Списокстудентоввстеке:\n");

for (int i = 0; i <= s->top; i++) {

printf("%s %d\n", s->data[i].lastName, s->data[i].priority);

}

}

elseif (choice == 4) {

break;

}

else {

printf("Неверныйвыбор!\n");

}

}

free(s->data);

free(s);

}

return 0;

}