Министерство образования Республики Беларусь

г. Минск

Государственное учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

**Лабораторная работа №5**

**“ПОНЯТИЕ СТЕКА. ОПЕРАЦИИ НАД СТЕКОМ”**

**Учебная группа 230501**

Выполнил: Кочеров Роман Сергеевич

Проверил: Селезнев Александр Игоревич

2023 год

**Вариант 4**

**Задание 1.** Создать стек для целых чисел. Максимальный размер стека вводится с экрана. Найти сумму после максимального элемента стека.

**Задание 2.** Создать два стека для целых чисел. Первый стек – организовать ввод по убыванию, второй стек – организовать ввод по возрастанию. Без сортировок и переворачивания исходных стеков сформировать третий стек упорядоченный по убыванию.

#include "Roma.h";

int main()

{

struct Stack s;

struct Stack s1;

struct Stack s2;

struct Stack s3;

int size1, size2;

printf("Task #1\n---------------------------------\n");

vvod\_stack(&s);

sumStack\_aftermax(&s);

printStack(&s); // Выводим элементы нового стека на экран

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

printf("Task #2\n---------------------------------\n");

printf("Enter elements on ascending\n");

size1 = vvod\_stack(&s1);

printf("Enter elements on descending\n");

size2 = vvod\_stack(&s2);

sort(&s1, &s2, &s3, size1, size2);

printf("Sort elements on descending\n");

printStack(&s3);

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

return 0;

}

#include <stdio.h>

#include <stdlib.h>

int vvod()

{

int var;

while ((scanf\_s("%d", &var) == 0) || getchar() != '\n')

{

printf("wrong input! try again\n");

rewind(stdin);

}

return var;

}

struct Stack

{

int\* data;

int top;

};

void push(struct Stack\* s, int value, int SIZE)

{

if (s->top == SIZE - 1) { // проверка на переполнение

printf("Error in add: steck is full\n");

exit(1);

}

s->top++;

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

}

int pop(struct Stack\* s)

{

if (s->top == -1) { // проверка на пустоту

printf("Error in delete: steck is empty\n");

exit(1);

}

int value = s->data[s->top];

s->top--;

return value;

}

int vvod\_stack(struct Stack\* s)

{

int maxSize;

printf("Enter the maximum stack size: ");

maxSize = vvod();

s->data = (int\*)calloc(maxSize, sizeof(int));

s->top = -1; // инициализация стека

// заполнение стека

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

int value;

printf("Enter elemnt #%d: ", i + 1);

value = vvod();

push(s, value, maxSize);

}

return maxSize;

}

void sumStack\_aftermax(struct Stack\* s)

{

int sumAfterMax = 0;

// поиск максимального элемента

int max = s->data[s->top];

int maxPos = s->top;

for (int i = s->top - 1; i >= 0; i--) {

if (s->data[i] > max) {

max = s->data[i];

maxPos = i;

}

}

// суммирование элементов после максимального

for (int i = maxPos + 1; i <= s->top; i++)

{

sumAfterMax += s->data[i];

}

printf("Sum of elements after maximum: %d\n", sumAfterMax);

}

void printStack(struct Stack\* s) {

printf("stack:\n");

for (int i = 0; i < s->top + 1; i++)

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

}

void sort(struct Stack\* s1, struct Stack\* s2, struct Stack\* s3, int size1, int size2)

{

int MAX\_SIZE, i;

int\* buf = NULL;

buf = (int\*)calloc(size2, sizeof(int));

if (buf == NULL)

{

printf("Error");

exit(1);

}

for (i = size2 - 1; i > -1; i--)

{

buf[i] = pop(s2);

}

MAX\_SIZE = size1 + size2;

s3->data = (int\*)calloc(MAX\_SIZE, sizeof(int));

s3->top = -1;

i = 0;

while (s1->top != -1 && i < size2)

{ // Пока оба стека не пусты

if (s1->data[s1->top] > buf[i]) // Если верхний элемент первого стека больше нижнего элемента второго стека

push(s3, pop(s1), MAX\_SIZE); // Добавляем верхний элемент первого стека в новый стек

else

{

push(s3, buf[i], MAX\_SIZE); // Иначе добавляем верхний элемент второго стека в новый стек

i++;

}

}

while (s1->top != -1) // Если первый стек еще не пустой, добавляем его оставшиеся элементы в новый стек

push(s3, pop(s1), MAX\_SIZE);

while (i < size2) // Если второй стек еще не пустой, добавляем его оставшиеся элементы в новый стек

{

push(s3, buf[i], MAX\_SIZE);

i++;

}

free(buf);

}



