Министерство науки и высшего образования РФ

ФГБОУ ВО Пермский национальный исследовательский политехнический университет

Кафедра «Вычислительной математики, механики и биомеханики»

Дисциплина «Информационные технологии»

Отчет по лабораторной работе

Тема: «Знакомство с языком программирования С №2»

Выполнил: студент группы ИСТ-20-1б Мясников Артем Евгеньевич

Проверил: преподаватель кафедры ВММБ Батин Сергей Евгеньевич

Пермь, 2021

# Упражнение 1.

Написать программу, которая генерирует массив из псевдослучайных целых чисел и выводит его в консоль в прямом порядке, и записывает в файл в обратном порядке.

Код:

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

int main(){

int r[5];

srand(time(NULL));

for(int i=0; i<5; i++){

r[i] = rand()%25;

printf("%d\n", r[i]);

}

FILE \*wr = fopen("out\_num\_1.txt", "w+");

for(int i=0; i<5; i++){

fprintf(wr, "%d\n", r[4-i]);

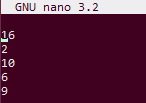
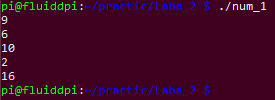
}

fclose(wr);

return 0;

}

Проверка:



Успешно.

# Упражнение 2.

Написать программу, которая принимает размер произвольного массива, создает его, заполняет случайными числами с плавающей точкой в интервале (0, 1). Выводит этот массив в консоль и записывает в файл в отсортированном виде.

Код:

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#define OUT\_FILE "output\_num\_2.txt"

float \* gen\_mass(int size){

float \*mass;

float r;

mass = malloc(size \* sizeof(float));

srand(time(NULL));

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

r = rand() %1000;

mass[i] = 1/r;

}

return mass;

}

void print\_mass(float\* mass, int size){

printf("\nRead\_in\_mass:\n");

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

printf("%f\n", mass[i]);

}

}

int main(int argc, char \*\*argv){

int m\_size = atoi(argv[1]);

float \*mass = gen\_mass(m\_size);

print\_mass(mass, m\_size);

for(int i=0; i<m\_size-1; i++) {

for(int j=0; j<m\_size-i-1; j++) {

if(mass[j] > mass[j+1]) {

float tmp = mass[j];

mass[j] = mass[j+1] ;

mass[j+1] = tmp;

}

}

}

FILE \*wr = fopen(OUT\_FILE, "w+");

for(int i=0; i<m\_size; i++){

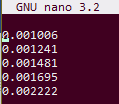
fprintf(wr, "%f\n", mass[i]);

}

fclose(wr);

}

Проверка



# Упражнение 3.

Создать структуру для хранения данных о названии и цене товара. Считать из входного файла данные о товарах. Написать функцию для сортировки товаров по цене. Вывести отсортированные данные в выходной файл.

Код:

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#define ERROR\_EXIT -1

struct product {

char name[64];

int price;

};

int main() {

FILE \*file\_input, \*file\_output;

char str[64];

char \*split\_str;

char \*estr;

int price;

char name[1028];

int count\_line;

printf("Открытие файла на чтение: ");

file\_input = fopen("input.txt", "r");

if (file\_input == NULL) {printf("ошибка\n"); return ERROR\_EXIT;}

else {printf("выполнено\n");}

struct product \*list;

list = malloc(256 \* (sizeof(char)\*64 + sizeof(int)));

printf("Считаны строки:\n");

split\_str = malloc(1028);

int i=0;

char \*spl = "-";

while (1) {

estr = fgets(str, sizeof(str), file\_input);

if (estr == NULL) {

if (feof(file\_input) != 0) {printf("\nЧтение файла закончено\n"); break;}

else {printf("\nОшибка чтения из файла\n"); break;}

}

split\_str = strtok(str, spl);

if (split\_str == NULL) {printf("error: strtok return NULL");}

strcpy(name, split\_str);

split\_str = strtok(NULL, spl);

price = atoi(split\_str);

strcpy(list[i].name, name);

list[i].price = price;

printf("\tname = %s\tprice = %d\n", list[i].name, list[i].price);

i++;

}

printf("Закрытие файла на чтение: ");

if (fclose(file\_input) == EOF) {printf("ошибка\n");}

else {printf("выполнено\n\n");}

/\*

printf("\n\n%s\n\n",list[2].name);

printf("До сортировки:\n");

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

printf("\tz = %d\n", z);

printf("\tname = %s\tprice = %d\n", list[z].name, list[z].price);

}

\*/

count\_line = i;

for(int i=0; i<count\_line-1; i++){

for(int j=0; j<count\_line - i - 1; j++){

if(list[j].price > list[j+1].price){

int tmp\_price = list[j].price;

char tmp\_name[64];

strcpy(tmp\_name, list[j].name);

list[j].price = list[j+1].price;

strcpy(list[j].name, list[j+1].name);

list[j+1].price = tmp\_price;

strcpy(list[j+1].name, tmp\_name);

}

}

}

printf("Открытие файла на запись: ");

file\_output = fopen("out\_num\_3.txt", "w");

if(file\_output == NULL) {printf("ошибка"); return ERROR\_EXIT;}

else {printf("успешно\n");}

printf("Запись в файл: \n");

char \*write = malloc(1024);

for(int i=0; i<count\_line ; i++){

strcpy(write, list[i].name);

fprintf(file\_output, "%s\t-\t%d\n", write, list[i].price);

printf("\tстрока %d записана\n", i);

}

printf("\nЗапись завершена\n");

printf("Закрытие файла: ");

if(fclose(file\_output) == EOF) {printf("ошибка\n");}

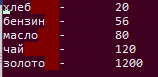
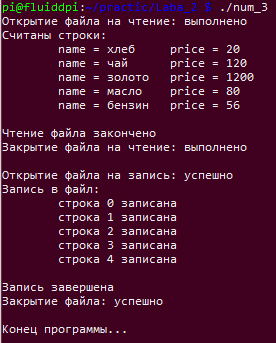
else {printf("успешно\n");}

printf("\nКонец программы...\n\n");

return 0;

}

Проверка:



# Упражнение 4.

Реализовать набор функций по работе со стеком:

* Организовать хранение стека в памяти.
* Реализовать добавление значения в стек.
* Реализовать извлечение значения из стека.
* Реализовать просмотр верхнего значения стека.
* Учесть при работе возможность переполнения стека.

Код:

#include <stdlib.h>

#include <stdio.h>

#define STACK\_MAX\_SIZE 20

#define STACK\_OVERFLOW -100

#define STACK\_UNDERFLOW -101

typedef int T;

typedef struct Stack\_tag {

T data[STACK\_MAX\_SIZE];

size\_t size;

} Stack\_t;

void push(Stack\_t \*stack, const T value){

if(stack->size >= STACK\_MAX\_SIZE) {

printf("Error: STACK\_OVERFLOW\n");

exit(STACK\_OVERFLOW);

}

stack->data[stack->size] = value;

stack->size++;

}

T pop(Stack\_t \*stack){

if(stack->size == 0) {

printf("Error: STACK\_UNDERFLOW\n");

exit(STACK\_UNDERFLOW);

}

stack->size--;

return stack->data[stack->size];

}

T peek(const Stack\_t \*stack){

if(stack->size <= 0) {

printf("Error: STACK\_UNDERFLOW\n");

exit(STACK\_UNDERFLOW);

}

return stack->data[stack->size-1];

}

void printStackValue(const T value) {printf("%d", value);}

void printStack(const Stack\_t \*stack){

int i;

int len = stack->size-1;

printf("stack %d > ", stack->size);

for(i=0; i<len; i++){

printStackValue(stack->data[i]);

printf("|");

}

if(stack->size != 0) {printStackValue(stack->data[i]);}

printf("\n");

}

int main(){

Stack\_t stack;

stack.size = 0;

printf("Введите размер стека: ");

int size\_stack;

scanf("%d", &size\_stack);

for(int i=0; i<size\_stack; i++){

push(&stack, i);

printStack(&stack);

}

printf("\nВерхний элемент: %d\n", peek(&stack));

printf("\nСостояние стека:\n");

printStack(&stack);

printf("\nВзяли верхний элемент: %d\n", pop(&stack));

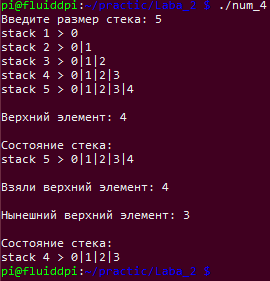
printf("\nНынешний верхний элемент: %d\n", peek(&stack));

printf("\nСостояние стека:\n");

printStack(&stack);

}

Проверка:



# Упражнение 5.

Написать программу, вычисляющую значение выражения, записанного в постфиксной (обратной польской) записи, считываемого из входного файла. Считать, что выражение может содержать только цифры и знаки «+», «-», «\*» и «/».

Код:

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#define FILE\_NAME "input\_num\_5.txt"

#define STACK\_MAX\_SIZE 100

#define ERROR\_STACK\_OVERFLOW -100

#define ERROR\_STACK\_UNDERFLOW -101

char\* readFile(){

char \*buffer = NULL;

int str\_size, read\_size;

FILE \*file = fopen(FILE\_NAME, "r");

if(file){

fseek(file, 0, SEEK\_END);

str\_size = ftell(file);

rewind(file);

buffer = (char\*) malloc(str\_size+1);

read\_size = fread(buffer, sizeof(char), str\_size, file);

buffer[str\_size] = '\0';

if(str\_size != read\_size){

free(buffer);

buffer = NULL;

}

fclose(file);

}

return buffer;

}

typedef int T;

typedef struct Stack\_tag {

T data[STACK\_MAX\_SIZE];

size\_t size;

} Stack\_t;

void push(Stack\_t \*stack, const T value){

if(stack->size >= STACK\_MAX\_SIZE) {

printf("Error\_push: STACK\_OVERFLOW\n");

exit(ERROR\_STACK\_OVERFLOW);

}

stack->data[stack->size] = value;

stack->size++;

}

T pop(Stack\_t \*stack){

if(stack->size == 0) {

printf("Error\_pop: STACK\_UNDERFLOW\n");

exit(ERROR\_STACK\_UNDERFLOW);

}

stack->size--;

return stack->data[stack->size];

}

T peek(const Stack\_t \*stack){

if(stack->size <= 0) {

printf("Error\_peek: STACK\_UNDERFLOW\n");

exit(ERROR\_STACK\_UNDERFLOW);

}

return stack->data[stack->size-1];

}

void printStackValue(const T value){

printf("%d", value);

}

void printStack(const Stack\_t \*stack){

int i;

int len = stack->size-1;

printf("stack %d > ", stack->size);

for(i=0; i<len; i++){

printStackValue(stack->data[i]);

printf("|");

}

if(stack->size != 0) {printStackValue(stack->data[i]);}

printf("\n");

}

int main(){

char \*text = readFile();

printf("Input:\n\t%s\n", text);

int len\_text = strlen(text);

Stack\_t stack;

stack.size=0;

for(int i=0; i<len\_text; i++){

if(isdigit(text[i])) {push(&stack, (int)(text[i]-'0'));}

if(text[i]=='+' || text[i]=='-' || text[i]=='\*' || text[i]=='/'){

int b = pop(&stack);

int a = pop(&stack);

switch(text[i]){

case '+' : {push(&stack, a+b);break;}

case '-' : {push(&stack, a-b);break;}

case '\*' : {push(&stack, a\*b);break;}

case '/' : {push(&stack, a/b);break;}

}

}

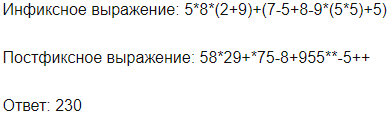
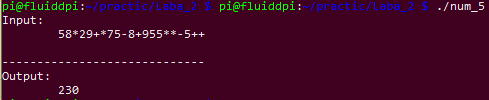
}

printf("-----------------------------\nOutput:\n\t%d\n", pop(&stack));

return 0;

}

Проверка:



# Упражнение 6.

Написать программу, реализующую перевод инфиксной записи арифметического выражения в постфиксную. Исходное арифметическое выражение состоит из цифр, знаков «+», «-», «\*», «/» и скобок.

Код:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define FILE\_NAME "input\_num\_6.txt"

#define QUEUE\_MAX\_SIZE 100

#define STACK\_MAX\_SIZE 100

#define QUEUE\_UNDERFLOW -101

#define STACK\_UNDERFLOW -101

#define TRUE 1

#define FALSE 0

/\*-----------------------------------------------------------------------------------------------\*/

/\* Функция для принятия данных из файла FILE\_NAME \*/

char\* readFile(){

char \*buffer = NULL;

int str\_size, read\_size;

FILE \*file = fopen(FILE\_NAME, "r");

if(file){

fseek(file, 0, SEEK\_END);

str\_size = ftell(file);

rewind(file);

buffer = (char\*) malloc(str\_size+1);

read\_size = fread(buffer, sizeof(char), str\_size, file);

buffer[str\_size] = '\0';

if(str\_size != read\_size){

free(buffer);

buffer = NULL;

}

fclose(file);

}

return buffer;

}

/\*################################################################\*/

/\* Очередь типа char \*/

/\* Тут все работает \*/

typedef char Type\_queue;

typedef struct Queue\_tag {

Type\_queue data[QUEUE\_MAX\_SIZE];

size\_t size;

} Queue\_t;

int pushQ(Queue\_t \*queue, const Type\_queue value){

/\*

Функиция для добавления элемента в очередь

В конец queue добавляется элемент value,

размер (queue->size) увеличивается на 1

Если максимальный размер (QUEUE\_MAX\_SIZE) достигнут,

то фунция ничего не запишет и вернет FALSE

Если элемент добавлен в очередь, функция вернет TRUE

\*/

if(queue->size >= QUEUE\_MAX\_SIZE) {

printf("Error: QUEUE\_OVERFLOW\n");

return FALSE;

}

else{

queue->data[queue->size] = value;

queue->size++;

return TRUE;

}

}

Type\_queue popQ(Queue\_t \*queue){

/\*

Фунция для изъятия одного элемента из очереди

Достается первый в очереди элемент, все остальные сдвигаются на 1 позицию к началу

Функция возвращает первый элемент

\*/

if(queue->size == 0){

printf("Error: QUEUE\_UNDERFLOW\n");

exit(QUEUE\_UNDERFLOW);

}

Type\_queue out = queue->data[0];

int len = queue->size;

for(int i=0; i<len; i++) {queue->data[i] = queue->data[i+1];}

queue->size--;

return out;

}

Type\_queue peekQ(Queue\_t \*queue){

if(queue->size == 0){

printf("Error: QUEUE\_UNDERFLOW\n");

exit(QUEUE\_UNDERFLOW);

}

return queue->data[0];

}

/\* Вывод одного переданного элемента \*/

void printQueueValue(const Type\_queue value) {printf("%c", value);}

void printQ(const Queue\_t \*queue){

/\*

Функция для вывода очереди в консоль

Формат:

queue (кол-во\_элементов\_в\_очереди) -> элемент\_1|элемнт\_2|элемент\_3|...|элемент\_n

\*/

int len = queue->size;

printf("queue %d -> ", queue->size);

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

printQueueValue(queue->data[i]);

if(i<len-1) {printf("|");}

else {printf("\n");}

}

}

/\*###################################################################\*/

/\* Стек типа char \*/

/\* Тут все работает \*/

typedef char T\_stack;

typedef struct Stack\_tag {

T\_stack data[STACK\_MAX\_SIZE];

size\_t size;

} Stack\_t;

/\* Функция добавления одного элемента на вершину стека \*/

int pushS(Stack\_t \*stack, const T\_stack value){

if(stack->size >= STACK\_MAX\_SIZE) {

printf("Error: STACK\_OVERFLOW\n");

return FALSE;

}

stack->data[stack->size] = value;

stack->size++;

return TRUE;

}

/\* Функция изъятия одного элемента из стека \*/

T\_stack popS(Stack\_t \*stack){

if(stack->size == 0) {

printf("Error\_pop: STACK\_UNDERFLOW\n");

exit(STACK\_UNDERFLOW);

}

stack->size--;

return stack->data[stack->size];

}

/\* Просмотр вершины стека \*/

T\_stack peekS(const Stack\_t \*stack){

if(stack->size <= 0) {

/\*printf("Error\_peek: STACK\_UNDERFLOW\n");\*/

}

return stack->data[stack->size-1];

}

/\* Вывод одного переданного элемента \*/

void printStackValue(const T\_stack value) {printf("%c", value);}

/\* Вывод всего переданного стека целиком \*/

void printS(const Stack\_t \*stack){

int i;

int len = stack->size-1;

printf("stack\_char %d > ", stack->size);

for(i=0; i<len; i++){

printStackValue(stack->data[i]);

printf("|");

}

if(stack->size != 0) {printStackValue(stack->data[i]);}

printf("\n");

}

/\*----------------------------------------------------------------------------------------------------------\*/

int main(){

char \*text = readFile();

printf("Input:\n\t%s\n", text);

Stack\_t stack;

stack.size = 0;

Queue\_t queue;

queue.size = 0;

int text\_len = strlen(text);

for(int i=0; i<text\_len; i++){

char top = peekS(&stack);

if(text[i]=='0' || text[i]=='1' || text[i]=='2' || text[i]=='3' || text[i]=='4' || text[i]=='5' || text$ pushQ(&queue, text[i]);

continue;

}

if(text[i]=='+' || text[i]=='-' || text[i]=='\*' || text[i]=='/' ){

if(stack.size==0){

pushS(&stack, text[i]);

continue;

}

if(top=='(') {

pushS(&stack, text[i]);

continue;

}

if((text[i]=='\*' || text[i]=='/') && (top=='\*' || top=='/')){

while((peekS(&stack)=='\*' || peekS(&stack)=='/') && stack.size!=0) {pushQ(&queue, popS($ pushS(&stack, text[i]);

continue;

}

if((text[i]=='\*' || text[i]=='/') && (top!='\*' || top!='/')) {

pushS(&stack, text[i]);

continue;

}

if((text[i]=='+' || text[i]=='-') && (top=='\*' || top=='/' || top=='+' || top=='-')){

while(peekS(&stack)!='(' && stack.size!=0) {pushQ(&queue, popS(&stack));}

pushS(&stack, text[i]);

continue;

}

}

if(text[i]=='(') {

pushS(&stack, text[i]);

continue;

}

if(text[i]==')') {

while(peekS(&stack)!='(' && stack.size!=0) {pushQ(&queue, popS(&stack));}

popS(&stack);

continue;

}

}

while(stack.size!=0) {pushQ(&queue, popS(&stack));}

/\*printQ(&queue);\*/

int out\_text\_len = queue.size;

char \*out\_text = malloc(out\_text\_len);

for(int i=0; i<out\_text\_len; i++){

out\_text[i] = popQ(&queue);

}

printf("--------------------------------------------\nOutput:\n\t%s\n", out\_text);

return 0;

}

Проверка:

