Скобелкин Глеб ФИб-1301

Лабораторная работа №3.

№1.

#include <iostream>

#define SIZE 5

void pop1(int value);

int push1();

void pop2(int value);

int push2();

bool empty();

bool full();

void printArray();

int arr[SIZE];

int i;

int main()

{

int choice1, choice2;

int item;

puts("Initial massive:");

printArray();

puts("\tWhat realization do you want to test?"

"\tEnter 1 or 2\n");

std::cout << "? ";

std::cin >> choice1;

switch (choice1) {

case 1:

puts("\tEnter 1 to pop.\n"

"\tEnter 2 to push.\n"

"\tEnter 3 to print.\n"

"\tEnter 4 to exit.\n");

std::cout << "? ";

std::cin >> choice2;

while (choice2 != 4) {

switch (choice2) {

case 1:

std::cout << "Enter the element: ";

std::cin >> item;

pop1(item);

break;

case 2:

push1();

break;

case 3:

printArray();

break;

default:

puts("Invalid choice. Try again.\n");

break;

}

std::cout << "? ";

std::cin >> choice2;

}

break;

case 2:

puts("\tEnter 1 to pop.\n"

"\tEnter 2 to push.\n"

"\tEnter 3 to print.\n"

"\tEnter 4 to exit.\n");

std::cout << "? ";

std::cin >> choice2;

while (choice2 != 4) {

switch (choice2) {

case 1:

std::cout << "Enter the element: ";

std::cin >> item;

pop2(item);

break;

case 2:

push2();

break;

case 3:

printArray();

break;

default:

puts("Invalid choice. Try again.\n");

break;

}

std::cout << "? ";

std::cin >> choice2;

}

break;

default:

puts("Error. Invalid choice\n");

break;

}

system("pause");

return 0;

}

void pop1(int value) {

arr[i++] = value;

}

int push1() {

int temp = arr[i--];

arr[i] = 0;

return temp;

}

void pop2(int value) {

for (int j = i - 1; j >= 0; --j)

arr[j + 1] = arr[j];

arr[0] = value;

++i;

}

int push2() {

int temp = arr[0];

for (int j = 1; j <= i; ++j)

arr[j - 1] = arr[j];

arr[i] = 0;

--i;

return temp;

}

bool empty() {

return false;

}

bool full() {

return false;

}

void printArray() {

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

std::cout << arr[i] << ' ';

puts("\n");

}

в первой реализации верхний элемент стека будет передвигаться в массиве вправо, а при добавлении новых элементов в стек предыдущие элементы будут оставаться в своих ячейках в массиве.

во второй реализации верхний элемент стека всегда будет находится в первой ячейке массива, а при добавлении новых элементов в стек предыдущие элементы будут сдвигаться на 1 ячейку вправо в массиве

№2.

#include <iostream>

#define SIZE 10

int arr[SIZE];

int i1;

int i2 = SIZE - 1;

void instructions();

void pop1(int value);

int push1();

bool empty1();

void pop2(int value);

int push2();

bool empty2();

bool full();

void printArray();

int main()

{

int choice;

int item;

puts("Initial massive:");

printArray();

instructions();

std::cout << "? ";

std::cin >> choice;

while (choice != 6) {

switch (choice) {

case 1:

std::cout << "Enter the element: ";

std::cin >> item;

pop1(item);

break;

case 2:

push1();

break;

case 3:

std::cout << "Enter the element: ";

std::cin >> item;

pop2(item);

break;

case 4:

push2();

break;

case 5:

printArray();

break;

default:

puts("Invalid choice.\n");

instructions();

break;

}

std::cout << "? ";

std::cin >> choice;

}

system("pause");

return 0;

}

void instructions() {

puts("Enter your choice:\n"

"\t1 to pop into the stack 1.\n"

"\t2 to push from the stack 1.\n"

"\t3 to pop into the stack 2.\n"

"\t4 to push from the stack 2.\n"

"\t5 to print array.\n"

"\t6 to end.");

}

void pop1(int value) {

if (!full())

arr[i1++] = value;

else

puts("Error. Stack is full.");

}

int push1() {

if (!empty1()) {

int temp = arr[i1--];

arr[i1] = 0;

return temp;

}

else

puts("Error. Stack is empty.");

}

bool empty1() {

if (i1 == 0)

return true;

else

return false;

}

void pop2(int value) {

if (!full())

arr[i2--] = value;

else

puts("Error. Stack is full.");

}

int push2() {

if (!empty2()) {

int temp = arr[i2++];

arr[i2] = 0;

return temp;

}

else

puts("Error. Stack is empty.");

}

bool empty2() {

if (i2 == SIZE - 1)

return true;

else

return false;

}

bool full() {

if (i1 > i2)

return true;

else

return false;

}

void printArray() {

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

std::cout << arr[i] << ' ';

puts("\n");

}

первый стек находится в начале массива, запись начинается с первой ячейки массива и идет "вправо".

// второй стек находится в конце массива, запись начинается с последней ячейки массива и идет "влево".

// проверка заполненности стека определяется тем, что вершины стеков находятся "по соседству".

№3.

#include <iostream>

#include <string>

bool inverted();

int main()

{

if (inverted())

std::cout << "TRUE" << std::endl;

else

std::cout << "FALSE" << std::endl;

system("pause");

return 0;

}

bool inverted() {

std::string s;

char c;

while ((c = getchar()) != '\*')

s += c;

for (int j = 1; j <= s.length(); ++j) {

if (s[s.length() - j] != getchar())

return false;

}

return true;

}

№4.

#include <iostream>

#include <string>

#include <stack>

std::string converting(std::string inf);

int main()

{

std::cout << converting("(1.1+2.2)\*(3.3+4.4)")

// (1+2)\*(3+4) | (11+24)\*(13+45) | 3.1\*6 |

// (1.1+2.2)\*(3.3+4.4)

<< std::endl;

system("pause");

return 0;

}

std::string converting(std::string inf) {

std::string prefix;

std::stack <char> stack;

int i = 0;

std::string w;

while (i < inf.length()) {

if (!isspace(inf[i])) {

if (inf[i] != '+' && inf[i] != '-' && inf[i] != '\*'

&& inf[i] != '/' && inf[i] != '^' && inf[i] != '('

&& inf[i] != ')') {

prefix += inf[i];

}

else if (inf[i] == '+' || inf[i] == '-' || inf[i] == '\*'

|| inf[i] == '/' || inf[i] == '^' || inf[i] == '(') {

stack.push(inf[i]);

prefix += ' ';

}

else if (inf[i] == ')') {

w = stack.top();

stack.pop();

while (w != "(") {

prefix += ' ';

prefix += w;

w = stack.top();

stack.pop();

}

}

}

++i;

}

while (!stack.empty()) {

prefix += ' ';

w = stack.top();

stack.pop();

prefix += w;

prefix += ' ';

}

return prefix;

}

заменил тип w с символа на строку, чтобы можно было в нем хранить не только цифру, но и число

№5.

#include <iostream>

#include <string>

#include <stack>

int main()

{

std::stack <int> stack;

std::string string;

std::string substr;

int i = 0;

int op1, op2;

puts("Enter the postfix form.");

getline(std::cin, string);

while (string.length() > i) {

if (!isspace(string[i])) {

if (isdigit(string[i])) {

while (isdigit(string[i]))

substr += string[i++];

stack.push(atoi(substr.c\_str()));

substr = "";

}

if (!isdigit(string[i]) && !isspace(string[i])) {

op1 = stack.top();

stack.pop();

op2 = stack.top();

stack.pop();

if (string[i] == '+')

stack.push(op2 + op1);

else if (string[i] == '-')

stack.push(op2 - op1);

else if (string[i] == '\*')

stack.push(op2 \* op1);

else if (string[i] == '/')

stack.push(op2 / op1);

else

stack.push(pow(op2, op1));

}

}

++i;

}

std::cout << "\nResult: " << stack.top() << '\n' << std::endl;

system("pause");

return 0;

}

операция ^ возведения в степень добавлена успешно!

№7.

#include <iostream>

#include <string>

#include <stack>

bool brackets(std::string string);

int main()

{

std::string string;

puts("Enter the string.");

getline(std::cin, string);

if (brackets(string))

puts("\nTRUE\n");

else

puts("\nFALSE\n");

system("pause");

return 0;

}

bool brackets(std::string string) {

std::stack <char> open;

for (int i = 0; i < string.length(); ++i) {

if (string[i] == '(' || string[i] == '[' || string[i] == '{')

open.push(string[i]);

else if (string[i] == ')' || string[i] == ']' || string[i] == '}') {

if (open.empty())

return false;

else if (string[i] == ')' && open.top() == '(' ||

string[i] == ']' && open.top() == '[' ||

string[i] == '}' && open.top() == '{')

open.pop();

else

return false;

}

}

if (!open.empty())

return false;

return true;

}

№8.

#include <iostream>

#define SIZE 10

int arr[SIZE];

int head;

int count;

int pop();

void push(int value);

bool empty();

bool full();

void instructions();

void printArray();

int main()

{

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

arr[i] = -9999;

int choice;

int item;

instructions();

std::cout << "? ";

std::cin >> choice;

while (choice != 4) {

switch (choice) {

case 1:

std::cout << "Enter the element: ";

std::cin >> item;

push(item);

break;

case 2:

pop();

break;

case 3:

printArray();

break;

}

std::cout << "? ";

std::cin >> choice;

}

system("pause");

return 0;

}

void instructions() {

puts("Enter your choice:\n"

"\t1 to push into the queue.\n"

"\t2 to pop from the queue.\n"

"\t3 to print array.\n"

"\t4 to end.");

}

void printArray() {

for (int i = head; i < SIZE; ++i)

if (arr[i] != -9999)

std::cout << arr[i] << ' ';

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

if (arr[i] != -9999)

std::cout << arr[i] << ' ';

puts("");

}

int pop() {

if (!empty()) {

int temp = arr[head];

arr[head] = -9999;

head = ++head % SIZE;

--count;

return temp;

}

puts("Error. Queue is empty.");

return -1;

}

void push(int value) {

if (!full())

arr[(head + count++) % SIZE] = value;

else

puts("Error. Queue is full.");

}

bool empty() {

return count == 0;

}

bool full() {

return count == 10;

}

очередь пуста, если количество элементов в очереди = 0

очередь полная, если количество элементов в очереди = размеру массива

№9.

#include <iostream>

#include <string>

#include <fstream>

#include <queue>

int main()

{

std::queue <int> queue1;

std::queue <int> queue2;

std::string string;

int a, b;

std::cout << "Enter A and B: ";

std::cin >> a >> b;

getchar();

puts("Fill the first file:");

getline(std::cin, string);

std::ofstream fout("file1.txt");

fout << string;

fout.close();

fout.open("file2.txt");

std::ifstream fin("file1.txt");

int number;

while (fin >> number) {

if (number < a)

fout << number << ' ';

else if (number >= a && number <= b)

queue1.push(number);

else

queue2.push(number);

}

fin.close();

while (!queue1.empty()) {

fout << queue1.front() << ' ';

queue1.pop();

}

while (!queue2.empty()) {

fout << queue2.front() << ' ';

queue2.pop();

}

fout.close();

system("pause");

return 0;

}

№10.

Функция Counter работает корректно. Логика прописана абсолютно верно, + проверил работоспособность в программе.

#include <iostream>

#include <ctime>

struct Tree {

int data;

Tree\* left;

Tree\* right;

};

int counter(Tree\* tree);

void insTree(Tree\* &tree, int value);

void printTree(Tree\* &root);

void del(Tree\* &root);

int main()

{

int N;

int item;

Tree\* root = NULL;

std::cout << "Enter the N: ";

std::cin >> N;

puts("\nINITIAL DATA:");

srand(time(NULL));

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

item = rand() % 10;

std::cout << item << ' ';

insTree(root, item);

}

puts("\n");

puts("TREE:");

printTree(root);

puts("\n");

std::cout << "Count: " << counter(root) << '\n' << std::endl;

std::cout << "Enter the element: ";

std::cin >> item;

std::cout << "treeSearch: " << treeSearch(root, item) << '\n' << std::endl;

std::cout << "treeMin: " << treeMin(root) << '\n' << std::endl;

del(root);

system("pause");

return 0;

}

int counter(Tree\* tree) { // 10

if (tree != NULL) {

if (tree->left == NULL && tree->right == NULL)

return 1;

else

return counter(tree->left) + counter(tree->right);

}

else

return 0;

}

№11.

Функция tree\_Search возвращает элемент, если он есть в дереве, иначе возвращает 0

№12.

Функция tree\_Min возвращает минимальный элемент в дереве

№13.

Функция tree\_Max возвращает максимальный элемент в дереве

№14.

#include <iostream>

#include <ctime>

struct Tree {

int data;

Tree\* leftPtr;

Tree\* rightPtr;

};

void insTree(Tree\* &tree, int value);

void printTree(Tree\* &root);

void del(Tree\* &root);

int treePred(Tree\* tree, int value);

int main()

{

int N;

int item;

Tree\* root = NULL;

std::cout << "Enter the N: ";

std::cin >> N;

puts("\nINITIAL DATA:");

srand(time(NULL));

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

item = rand() % 9 + 1;

std::cout << item << ' ';

insTree(root, item);

}

puts("\n");

puts("TREE:");

printTree(root);

puts("\n");

std::cout << "Enter the element: ";

std::cin >> item;

item = treePred(root, item);

if (item != 0)

std::cout << "\nPrevious element is: " << item << '\n' << std::endl;

else

std::cout << "\nError. Element not found or you wrote root-element\n" << std::endl;

del(root);

system("pause");

return 0;

}

void insTree(Tree\* &tree, int value) {

if (tree == NULL) {

tree = new Tree;

tree->leftPtr = NULL;

tree->rightPtr = NULL;

tree->data = value;

}

else if (value <= tree->data) {

insTree(tree->leftPtr, value);

}

else {

insTree(tree->rightPtr, value);

}

}

void printTree(Tree\* &root) {

if (root != NULL) {

printTree(root->leftPtr);

std::cout << root->data << ' ';

printTree(root->rightPtr);

}

}

void del(Tree\* &root) {

if (root != NULL) {

del(root->leftPtr);

del(root->rightPtr);

delete root;

root = NULL;

}

}

int treePred(Tree\* tree, int value) {

int previousElem = 0;

while (tree != NULL && tree->data != value) {

previousElem = tree->data;

if (value < tree->data)

tree = tree->leftPtr;

else

tree = tree->rightPtr;

}

if (tree != NULL)

return previousElem;

else

return 0;

}

№15.

#include <iostream>

#include <ctime>

struct Tree {

int data;

Tree\* leftPtr;

Tree\* rightPtr;

};

void insTree(Tree\* &tree, int value);

void printTree(Tree\* &root);

void del(Tree\* &root);

Tree\* secondMinPtr(Tree\* tree);

int main()

{

int N;

int item;

Tree\* root = NULL;

std::cout << "Enter the N: ";

std::cin >> N;

puts("\nINITIAL DATA:");

srand(time(NULL));

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

item = rand() % 9 + 1;

std::cout << item << ' ';

insTree(root, item);

}

puts("\n");

puts("TREE:");

printTree(root);

puts("\n");

std::cout << "SecondMinPtr: " << secondMinPtr(root) << '\n' << std::endl;

std::cout << '\t' << secondMinPtr(root)->data << '\n' << std::endl;

del(root);

system("pause");

return 0;

}

void insTree(Tree\* &tree, int value) {

if (tree == NULL) {

tree = new Tree;

tree->leftPtr = NULL;

tree->rightPtr = NULL;

tree->data = value;

}

else if (value <= tree->data) {

insTree(tree->leftPtr, value);

}

else {

insTree(tree->rightPtr, value);

}

}

void printTree(Tree\* &root) {

if (root != NULL) {

printTree(root->leftPtr);

std::cout << root->data << ' ';

printTree(root->rightPtr);

}

}

void del(Tree\* &root) {

if (root != NULL) {

del(root->leftPtr);

del(root->rightPtr);

delete root;

root = NULL;

}

}

Tree\* secondMinPtr(Tree\* tree) {

if (tree != NULL) {

if (tree->leftPtr == NULL && tree->rightPtr != NULL)

return tree;

else if (tree->leftPtr == NULL && tree->rightPtr == NULL) {

puts("ERROR.");

system("pause");

exit(1);

}

else {

Tree\* previousMinPtr = tree;

while (tree->leftPtr != NULL) {

previousMinPtr = tree;

tree = tree->leftPtr;

}

return previousMinPtr;

}

}

}

№16.

#include <iostream>

#include <ctime>

struct Tree {

int data;

Tree\* leftPtr;

Tree\* rightPtr;

};

void insTree(Tree\* &tree, int value);

void printTree(Tree\* &root);

void del(Tree\* &root);

int idealBalance(Tree\* tree);

int main()

{

int N;

int item;

Tree\* root = NULL;

std::cout << "Enter the N: ";

std::cin >> N;

puts("\nINITIAL DATA:");

srand(time(NULL));

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

item = rand() % 9 + 1;

std::cout << item << ' ';

insTree(root, item);

}

puts("\n");

puts("TREE:");

printTree(root);

puts("\n");

if (idealBalance(root) != 0)

std::cout << "Tree ideal balanced.\n" << std::endl;

else

std::cout << "Tree NOT ideal balanced.\n" << std::endl;

del(root);

system("pause");

return 0;

}

void insTree(Tree\* &tree, int value) {

if (tree == NULL) {

tree = new Tree;

tree->leftPtr = NULL;

tree->rightPtr = NULL;

tree->data = value;

}

else if (value <= tree->data) {

insTree(tree->leftPtr, value);

}

else {

insTree(tree->rightPtr, value);

}

}

void printTree(Tree\* &root) {

if (root != NULL) {

printTree(root->leftPtr);

std::cout << root->data << ' ';

printTree(root->rightPtr);

}

}

void del(Tree\* &root) {

if (root != NULL) {

del(root->leftPtr);

del(root->rightPtr);

delete root;

root = NULL;

}

}

int idealBalance(Tree\* tree) {

if (tree == NULL)

return 1;

int left = idealBalance(tree->leftPtr);

int right = idealBalance(tree->rightPtr);

if (left == 0 || right == 0)

return 0;

if (abs(left - right) > 1)

return 0;

if (left > right)

return left + 1;

else

return right + 1;

}

// обхожу все дерево. если дошел до конца поддерева, возвращаю 1. далее возвращаю максимальное значение правой и левой глубины.

// во время обхода проверяю, если правая и левая глубины различаются более чем на 1, то возвращаю 0.

// если функция возвращает 0, значит дерево не идеально сбалансированное.