Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

кафедра «Вычислительных методов и программирования»

**ОТЧЁТ**

По лабораторным работам №7, №8, №9

«СТЕКИ. ДВУСВЯЗНЫЕ СПИСКИ.

ДРЕВОВИДНЫЕ СТРУКТУРЫ ДАННЫХ»

Выполнил:

Студент ФИТиУ

гр. 020601

Гудков А. С.

Вариант №7

Проверил:

ассистент кафедры ВМиП

Беспалов С. А.

Минск 2021

***Цель работы:***

Научиться реализовывать и работать с динамическими структурами данных, такими как стеки, двусвязные списки и деревья. Написать и отладить программу для работы с динамической структурой данных.

***Индивидуальное задание (ЛР №7):***

*Создать стек с числами в диапазоне от –50 до +50. В конце работы все стеки должны быть удалены.*

7. Определить, сколько элементов стека, начиная с вершины, находится до элемента с максимальным значением.

***Индивидуальное задание (ЛР №8):***

*Создать двусвязный список, состоящий из n целых чисел. Информационную часть в оперативной памяти не перемещать. В конце работы освободить всю динамически выделенную память.*

7. Поместить во второй список элементы, находящиеся между максимальным и минимальным элементами первого списка.

***Индивидуальное задание (ЛР №9):***

*Создать сбалансированное дерево поиска и распечатать информацию прямым, обратным обходом и в порядке возрастания. Написать функции добавления нового значения, удаления значения, поиска значения. Вся выделенная память должна быть освобождена.*

7. Определить количество символов во всех строках, находящихся в узлах дерева.

***Текст программы (ЛР №7):***

#include <iostream>

using namespace std;

struct **Node**

{

int inf;

Node\* next;

};

struct **Stack**

{

Node\* top = nullptr;

void push\_back(int inf)

{

Node\* spt = new Node;

spt->inf = inf;

spt->next = top;

top = spt;

}

void show()

{

Node\* spt = top;

while (spt)

{

cout << spt->inf << ' ';

spt = spt->next;

}

cout << endl;

}

void clear()

{

Node\* spt;

while (top)

{

spt = top;

top = top->next;

delete spt;

}

top = nullptr;

}

};

int **main**()

{

int n;

cout << "Enter stack size: "; cin >> n;

if (n > 0)

{

// create

**Stack** s;



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

s.push\_back(rand() % 101 - 50);

cout << "Stack: ";

s.show();

// find

Node\* smax = s.top;

for (Node\* spt = s.top->next; spt != nullptr; spt = spt->next)

if (smax->inf < spt->inf) smax = spt;

cout << "Max element: " << smax->inf << endl;

n = 0;

for (Node\* spt = s.top; spt != smax; spt = spt->next) n++;

cout << "Elements before max: " << n << endl;

// clear

s.clear();

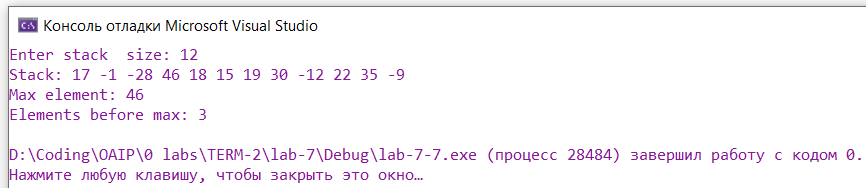
}

else cout << "Stack is empty" << endl;

return 0;

}

***Результат работы программы:***





***Текст программы (ЛР №8):***

#include <iostream>

using namespace std;

struct **Node**

{

int inf;

Node\* prev;

Node\* next;

};

struct **List**

{

Node\* begin = nullptr;

Node\* end = nullptr;

void push\_back(int inf)

{

Node\* spt = new Node;

spt->inf = inf;

spt->next = nullptr;

if (begin) { spt->prev = end; end->next = spt; end = spt; }

else {spt->prev = nullptr; begin = end = spt; }

}

void show()

{

Node\* spt = begin;

while (spt)

{

cout << spt->inf << ' ';

spt = spt->next;

}

cout << endl;

}

void clear()

{

Node\* spt;

while (begin)

{

spt = begin;

begin = begin->next;

delete spt;

}

begin = end = nullptr;

}

};

int **main**()

{

int n;

cout << "Enter list size: "; cin >> n;

if (n > 0)

 {

// create

**List** l1;

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

l1.push\_back(rand() % 101 - 50);

cout << "List: ";

l1.show();

// find

Node\* sp1, \*sp2;

int i, i1, i2;

sp1 = l1.begin;

i1 = i = 0;

for (Node\* spt = l1.begin->next; spt; spt = spt->next, i++)

if (spt->inf < sp1->inf) { sp1 = spt; i1 = i; }

cout << "Min: " << sp1->inf << endl;

sp2 = l1.begin;

i2 = i = 0;

for (Node\* spt = l1.begin->next; spt; spt = spt->next, i++)

if (spt->inf > sp2->inf) { sp2 = spt; i2 = i; }

cout << "Max: " << sp2->inf << endl;

// create new list

**List** l2;

Node\* begin = (i1 < i2) ? sp1 : sp2;

Node\* end = (i1 < i2) ? sp2 : sp1;

for (Node\* spt = begin->next; spt && spt != end; spt = spt->next)

l2.push\_back(spt->inf);

cout << "New List: ";

l2.show();

// clear

l1.clear();

l2.clear();

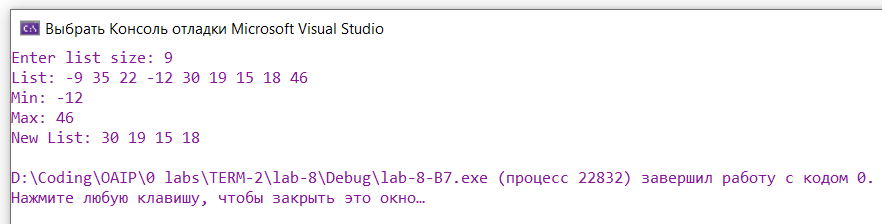
}

else cout << "List is empty";

return 0;

}

***Результат работы программы:***



***Текст программы (ЛР №9):***

#include <iostream>

#include <string.h>

using namespace std;

struct **Node**

{

char inf[20];

Node\* left;

Node\* right;

};

struct **Tree**

{

Node\* root = nullptr;

void push(char\* inf)

{

Node\* pt = new Node;

strcpy\_s(pt->inf, inf);

pt->left = pt->right = nullptr;

if (root == nullptr) root = pt;

else

{

Node\* r1, \* r2;

r1 = r2 = root;

while (r1)

{

r2 = r1;

if (strcmp(inf, r1->inf) == -1) r1 = r1->left;

else r1 = r1->right;

}

if (strcmp(inf, r2->inf) == -1) r2->left = pt;

else r2->right = pt;

}

}

void show\_grow(Node\* p)

{

if (p == nullptr) return;

show\_grow(p->left);

cout << p->inf << ' ';

show\_grow(p->right);

}

void show\_direct(Node\* p)

{

if (p == nullptr) return;

cout << p->inf << ' ';

show\_direct(p->left);

show\_direct(p->right);

}

void show\_reverse(Node\* p)

{

 if (p == nullptr) return;

show\_reverse(p->left);

show\_reverse(p->right);

cout << p->inf << ' ';

}

void clear(Node\* p)

{

if (p == nullptr) return;

clear(p->left);

clear(p->right);

delete p;

p = nullptr;

}

int str\_len(Node\* p)

{

if (p == nullptr) return 0;

else return strlen(p->inf) + str\_len(p->left) + str\_len(p->right);

}

};

int **main**()

{

int n;

cout << "Enter number of storkes: "; cin >> n;

if (n > 0)

{

// create

**Tree** t;

char str[20];

getchar();

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

{

cout << i+1 << ": "; cin >> str;

t.push(str);

}

cout << endl << "Tree (grow order): " << endl;

t.show\_grow(t.root);

cout << endl << "Tree (direct order): " << endl;

t.show\_direct(t.root);

cout << endl << "Tree (reverse order): " << endl;

t.show\_reverse(t.root);

// count

cout << endl << "Number of chars: " << t.str\_len(t.root) << endl;

// clear

t.clear(t.root);

}

else cout << "Tree is empty" << endl;

return 0;

}

***Результат работы программы:***

