**Учреждение образования**

**„Белорусский государственный университет информатики и радиоэлектроники”**

**Кафедра «Вычислительных методов и программирования»**

**ОТЧЕТ**

**По лабораторным работам №13** - **№15**

**«Стеки. Двухсвязные списки.**

**Древовидные структуры данных.»**

**Выполнил:**

**Студент АСОИ**

**Группа №020601**

**Шумигай В.В.**

**Вариант № 15**

**Проверил:**

**Беспалов С.A.**

**Минск 2021**

**Цель работы:** сформировать умения работы с односвязными и двухсвязными списками, древовидными структурами данных.

**Индивидуальное задание:**

**13.** Создать стек, состоящий из n целых чисел. Выполнить задание. Информационную часть в оперативной памяти не перемещать. Результат вывести на экран. В конце работы освободить всю динамически выделенную память.

Удалить из стека все элементы, значения которых находятся в диапазоне от 0 до 10.

**14.** Создать двусвязный список, состоящий из n символов латинского алфавита. Преобразовать его в два списка: первый список должен содержать прописные символы, второй − строчные.

**15.** Создать сбалансированное дерево поиска, состоящее из целых чисел. Вывести информацию на экран, используя прямой, обратный и симметричный обход дерева. Выполнить задание, результат вывести на экран. В конце работы освободить всю динамически выделенную память.

Удалить из дерева узел с заданным ключом.

**Текст программы 13:**

#include <iostream>

using namespace std;

struct node {

int info;

node\* next;

} \*t;

node\* instack(node\* p, int inf)

{

node\* t = new node;

t->info = inf;

t->next = p;

return t;

}

void view(node\* p)

{

node\* t = p;

while (t != nullptr) {

cout << t->info << " ";

t = t->next;

}

}

node\* outstack(node\* p, int\* out)

{

node\* t = p;

\*out = p->info;

cout << p->info << endl;

p = p->next;

delete t;

return p;

}

void delete\_all(node\*\* p)

{

node\* t;

while (\*p != nullptr) {

t = \*p;

\*p = (\*p)->next;

delete t;

}

}

int main() {

int n, inf;

cout << "Vvedite kolichestvo chisel: "; cin >> n; cout << endl;

int\* a = new int[n];

cout << "Vvedite chisla: " << endl;

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

{

cin >> inf;

a[i] = inf;

t = instack(t, inf);

}

view(t);

if (t != nullptr)

delete\_all(&t);

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

{

if (a[i] > 0 && a[i] < 10)

{

t = instack(t, a[i]);

}

}

cout << endl;

cout << "Noviy steck:" << endl;

view(t);

delete[]a;

delete\_all(&t);

return 0;

}

**Текст программы 14:**

#include <iostream>

using namespace std;

struct node {

char inf;

node\* next;

node\* prev;

}\*tmp = nullptr, \*Head = nullptr, \*Tail = nullptr;

void show(node\* tm, node\* Head)

{

tm = Head;

while (tm != nullptr) {

cout << tm->inf << " ";

tm = tm->next;

}

cout << "\n";

}

void fill\_list(int \*n)

{

char sim;

for (int i = 0; i < \*n; i++) {

tmp = new node;

cout << "Vvedite znach " << i + 1 << "-go elementa: "; cin >> sim;

tmp->inf = sim;

tmp->next = nullptr;

if (Head != nullptr) {

tmp->prev = Tail;

Tail->next = tmp;

Tail = tmp;

}

else {

tmp->prev = nullptr;

Head = Tail = tmp;

}

}

show(tmp, Head);

}

void delete\_list(node \*tm, node\* Head, node\* Tail)

{

node\* t; tm = Head;

while (tm != nullptr) {

t = tm;

tm = tm->next;

delete t;

}

Head = Tail = nullptr;

}

void preob()

{

node\* tms, \* Tails, \* Heads;

node\* tmb, \* Tailb, \* Headb;

tms = Heads = Tails = tmb = Headb = Tailb = nullptr;

tmp = Head;

while (tmp != nullptr) {

if (tmp->inf > 96 && tmp->inf < 123)

{

tms = new node;

tms->inf = tmp->inf;

tms->next = nullptr;

if (Heads != nullptr) {

tms->prev = Tails;

Tails->next = tms;

Tails = tms;}

else {

tms->prev = nullptr;

Heads = Tails = tms;}

}

else if (tmp->inf > 64 && tmp->inf < 91)

{

tmb = new node;

tmb->inf = tmp->inf;

tmb->next = nullptr;

if (Headb != nullptr) {

tmb->prev = Tailb;

Tailb->next = tmb;

Tailb = tmb;

}

else {

tmb->prev = nullptr;

Headb = Tailb = tmb;

}

}

tmp = tmp->next;

}

show(tms, Heads);

show(tmb, Headb);

delete\_list(tms, Heads, Tails);

delete\_list(tmb, Headb, Tailb);

}

int main()

{

int n;

cout << "Vvedite kolichestvo elementov: "; cin >> n;

fill\_list(&n);

preob();

delete\_list(tmp, Head, Tail);

return 0;

}

**Текст программы 15:**

#include <iostream>

using namespace std;

struct node {

int inf;

node\* left, \* right;

}\*tree = NULL;

void tree\_pram\_obxod(node\* tree, int\*mas1)

{

if (tree != NULL) {

cout << mas1[tree->inf-1] << " ";

tree\_pram\_obxod(tree->left, mas1);

tree\_pram\_obxod(tree->right, mas1);

}

}

void tree\_obr\_obxod(node\* tree, int\* mas1)

{

if (tree != NULL) {

tree\_obr\_obxod(tree->left, mas1);

tree\_obr\_obxod(tree->right, mas1);

cout << mas1[tree->inf-1] << " ";

}

}

void tree\_sim\_obxod(node\* tree, int\*mas1) {

if (tree != NULL) {

tree\_sim\_obxod(tree->left, mas1);

cout << mas1[tree->inf-1] << " ";

tree\_sim\_obxod(tree->right, mas1);

}

}

void print\_Tree(node\* p, int level, int\* mas1)

{

if (p)

{

print\_Tree(p->left, level + 1, mas1);

for (int i = 0; i < level; i++) cout << " ";

cout << mas1[p->inf-1] << endl;

print\_Tree(p->right, level + 1, mas1);

}

}

node\* addtree(int L, int R, int\* mas) {

int m; node\* tree;

if (L > R) return NULL;

m = (L + R) / 2;

tree = new node;

tree->inf = mas[m-1];

tree->left = addtree(L, m - 1, mas);

tree->right = addtree(m + 1, R, mas);

return tree;

}

node\* delete\_node(node\* tree, int val)

{

if (tree == NULL)

return tree;

if (val == tree->inf) {

node\* tmp;

if (tree->right == NULL)

tmp = tree->left;

else {

node\* ptr = tree->right;

if (ptr->left == NULL) {

ptr->left = tree->left;

tmp = ptr;

}

else {

node\* pmin = ptr->left;

while (pmin->left != NULL) {

ptr = pmin;

pmin = ptr->left;

}

ptr->left = pmin->right;

pmin->left = tree->left;

pmin->right = tree->right;

tmp = pmin;

}

}

delete tree;

return tmp;

}

else if (val < tree->inf) tree->left = delete\_node(tree->left, val);

else tree->right = delete\_node(tree->right, val);

return tree;

}

void ClearNode(node\* tree) {

if (tree != NULL) {

if (tree->left != NULL)

ClearNode(tree->left);

if (tree->right != NULL)

ClearNode(tree->right);

delete tree;}

}

int main()

{

int n, x;

cout << "Vvedite kolichestvo uzlov: "; cin >> n;

int \*mas = new int[n];

int\* mas1 = new int[n];

for (int i = 0; i < n; i++) mas[i] = i + 1;

cout << "Vvedite elementi: ";

for (int i = 0; i < n; i++) cin >> mas1[i];

for (int i = 0; i < n - 1; i++) {

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

if (mas1[i] < mas1[j]) swap(mas1[i], mas1[j]);

}

}

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

tree = addtree(mas[0], n, mas);

cout << "Derevo:" << endl;

print\_Tree(tree, 5, mas1); cout << endl;

cout << "Pramoy obxod: ";

tree\_pram\_obxod(tree, mas1); cout << endl;

cout << "Obratniy obxod: ";

tree\_obr\_obxod(tree, mas1); cout << endl;

cout << "Simmetrichniy obxod: ";

tree\_sim\_obxod(tree, mas1); cout << endl << endl;

int val; cout << "Vvedite elem dla udalenia: "; cin >> val;

for (int i = 0; i < n; i++) if (mas1[i] == val) val = i+1;

delete\_node(tree, val);

cout << "Derevo:" << endl;

print\_Tree(tree, n-1, mas1); cout << endl;

tree\_sim\_obxod(tree, mas1);

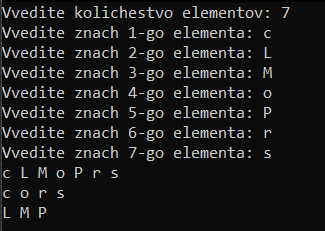
ClearNode(tree);

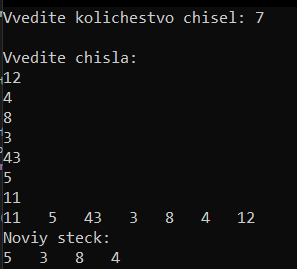
delete[] mas;

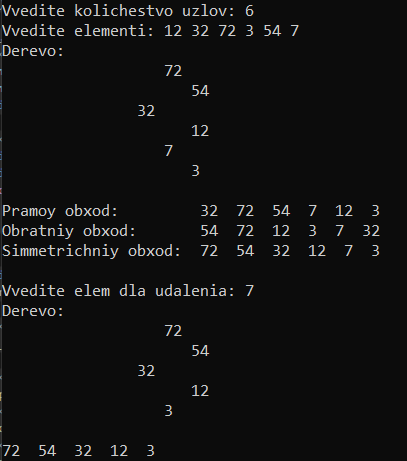
delete[] mas1;

return 0;

}**Результат работы программы:**

**13: 14:**

****

**15:**