**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ**

**Факультет непрерывного и дистанционного обучения**

# Специальность: Автоматизированные системы обработки информации

**ЛАБОРАТОРНЫЙ ПРАКТИКУМ**

**ПО ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ  
ПРОГРАММИРОВАНИЕ И ПРОЕКТИРОВАНИЕ № 1-4**

**Вариант № 4**

***Соболевского Дмитрия Александровича***

***Группа: 590651***

***Зачетная книжка: ‎000623-28***

***Электронный адрес:*** [***sobolevskidmitry@gmail.com***](mailto:sobolevskidmitry@gmail.com) ***/ BSUIR\sda***

**1. ИНДИВИДУАЛЬНЫЕ ЗАДАНИЯ**

        Для всех вариантов заданий:

* определить типы и функции в соответствии с вариантом задания;
* в функции **main()** реализовать демонстрацию работы созданных функций;

**ЗАМЕЧАНИЯ:**

        Весь ввод/вывод должен осуществляться только внутри функции **main()**. Если в функции необходимо передавать объекты типа структура, то передавать их следует через указатель либо ссылку на объект.

**4)** Имеется узел бинарного дерева:

struct Node

{

char name[10]; //имя узла

Node \* left; //левая ветвь

Node \* right; //правая ветвь

};

        Определите функцию добавления новых узлов в дерево **AddNode()** и функцию удаления дерева **DelTree()**.Определите рекурсивную функцию вывода такого дерева **PrintTree()** на экран.

**РЕШЕНИЕ:**

struct SNode

{

char name[10];

SNode \*left;

SNode \*right;

};

void AddNode(SNode \*node, const char \*data);

void DelTree(SNode \*tree);

void PrintTree(SNode \*tree);

void AddNode(SNode \*node, const char \*data){

static char buffer[12] = { 0 };

int i = 0;

// На случай, если длина строки равна 10

for (i = 0; i < 10 && node->name[i]; i++) buffer[i] = node->name[i];

buffer[i] = 0;

std::string s1(buffer), s2(data);

// Если вставляемая строка меньше текущей

if (s1 > s2){

if (!node->left){

SNode \*newnode = new SNode;

newnode->left = newnode->right = NULL;

int i = 0;

for (i = 0; i < 10 && data[i]; i++) newnode->name[i] = data[i];

if (i < 10) newnode->name[i] = 0;

node->left = newnode;

}

// Идти в левое поддерево

else{ AddNode(node->left, data); }

}

// Если вставляемая строка больше текущей

if (s1 < s2){

if (!node->right){

SNode \*newnode = new SNode;

newnode->left = newnode->right = NULL;

int i = 0;

for (i = 0; i < 10 && data[i]; i++) newnode->name[i] = data[i];

if (i < 10) newnode->name[i] = 0;

node->right = newnode;

}

// Идти в правое поддерево

else{ AddNode(node->right, data); }

}

}

void DelTree(SNode \*tree){

if (tree->left) DelTree(tree->left);

delete tree->left; tree->left = NULL;

if (tree->right) DelTree(tree->right);

delete tree->right; tree->right = NULL;

}

void PrintTree(SNode \*tree){

static char buffer[12] = { 0 };

int i = 0;

// Вывести имя

for (i = 0; i < 10 && tree->name[i]; i++) buffer[i] = tree->name[i];

buffer[i] = 0;

std::cout << buffer << std::endl;

// Идти в левое поддерево, если оно есть

if (tree->left) PrintTree(tree->left);

// Идти в правое поддерево, если оно есть

if (tree->right) PrintTree(tree->right);

}

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

SNode \*snode = new SNode;

// Лабораторная 1

snode->left = snode->right = 0;

strcpy\_s(snode->name, 10, "start");

snode->name[5] = 0;

setlocale(LC\_ALL, "RUSSIAN");

std::cout << "Лабораторная 1" << std::endl;

std::cout << "Состояние дерева до вставки" << std::endl;

PrintTree(snode);

AddNode(snode, "g4e3ee");

AddNode(snode, "2393w3w9");

AddNode(snode, "`9i3wdd");

std::cout << "Состояние дерева после вставки" << std::endl;

PrintTree(snode); }

**2. ИНДИВИДУАЛЬНЫЕ ЗАДАНИЯ**

        Для всех вариантов заданий:

* определить классы их свойства и методы в соответствии с вариантом задания;
* в функции **main()** реализовать демонстрацию работы с объектами созданных классов;

**ЗАМЕЧАНИЯ:**

**Варианты заданий**

        Весь ввод/вывод должен осуществляться только внутри функции **main()**.

        Передача в функции объектов типа класс должна осуществляться через указатель либо ссылку на объект.

        Все другие нестатические данные должны передаваться функциям через параметры.

        Все нестатические элементы класса должны иметь закрытый атрибут доступа. Доступ извне к таким элементам должен осуществляться через функции-элементы класса.

**4)** Определите узел бинарного дерева следующим образом:

class Node

{

char name[10]; //имя узла

Node \* left; //левая ветвь

Node \* right; //правая ветвь

...

};

        Определите в классе следующие функции:

**Init()** – инициализация узла. Функция должна установить указатели на левый и правый узел в ноль;

**AddNode()** – добавление узла в левую или правую ветви. Если слева нет узла, то добавить слева, если справа нет узла, то добавить справа, иначе ничего не добавлять;

**DelTree()** – удаление поддеревьев;

**Print()** – рекурсивная функция вывода дерева на экран.

**РЕШЕНИЕ:**

class Node

{

private:

char name[10];

Node \*left;

Node \*right;

bool isnull; // Нулевой объект

void Init();

public:

Node();// Пустой конструктор

void AddNode(const char \*name);

void DelTree();

void Print();

bool isNull();

};

void Node::Init(){

left = right = NULL;

isnull = true;

memset(name, 0, 10);

}

void Node::AddNode(const char \*name){

static char buffer[12] = { 0 };

int i = 0;

// На случай, если длина строки равна 10

for (i = 0; i < 10 && this->name[i]; i++) buffer[i] = this->name[i];

buffer[i] = 0;

std::string s1(this->name), s2(name);

// Если вставляемая строка меньше текущей

if (s1 > s2){

if (!this->left){

this->left = new Node(name);

}

// Идти в левое поддерево

else{ this->left->AddNode(name); }

}

// Если вставляемая строка больше текущей

if (s1 < s2){

if (!this->right){

this->right = new Node(name);

}

// Идти в правое поддерево

else{ this->right->AddNode(name); }

}

}

void Node::DelTree(){

if (this->left) this->left->DelTree();

this->left = NULL;

if (this->right) this->right->DelTree();

this->right = NULL;

void \*data = this;

delete data;

}

void Node::Print(){

static char buffer[12] = { 0 };

int i = 0;

if (this->isnull){

std::cout << "null" << std::endl; return; }

// Вывести имя

for (i = 0; i < 10 && this->name[i]; i++) buffer[i] = this->name[i];

buffer[i] = 0;

std::cout << buffer << std::endl;

// Идти в левое поддерево, если оно есть

if (this->left) this->left->Print();

// Идти в правое поддерево, если оно есть

if (this->right) this->right->Print();

}

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

std::cout << "Лабораторные 2 и 3" << std::endl;

std::cout << "Состояние дерева до вставки" << std::endl;

Node node("lab2");

node.Print();

node.AddNode("ty54rrtr");

node.AddNode("w33w0odr03w");

node.AddNode("3w032i3");

node.AddNode("node name");

std::cout << "Состояние дерева после вставки" << std::endl;

node.Print();

}

**3. ИНДИВИДУАЛЬНЫЕ ЗАДАНИЯ**

        Для всех вариантов заданий:

* определить классы их свойства и методы в соответствии с вариантом задания;
* в функции **main()** реализовать демонстрацию работы с объектами созданных классов;

**ЗАМЕЧАНИЯ:**

* Весь ввод/вывод должен осуществляться только внутри функции **main()**.
* Передача в функции объектов типа класс должна осуществляться через указатель либо ссылку на объект.
* Все другие нестатические данные должны передаваться функциям через параметры.
* Все нестатические элементы класса должны иметь закрытый атрибут доступа. Доступ извне к таким элементам должен осуществляться через функции-элементы класса.

**4)** Определите узел бинарного дерева следующим образом:

class Node

{

char name[10]; //имя узла

Node \* left; //левая ветвь

Node \* right; //правая ветвь

...

};

        Определите в классе следующие функции:

**Init()** – инициализация узла. Функция должна установить указатели на левый и правый узел в ноль;

**AddNode()** – добавление узла в левую или правую ветви. Если слева нет узла, то добавить слева, если справа нет узла, то добавить справа, иначе ничего не добавлять;

**DelTree()** – удаление поддеревьев;

**Print()** – рекурсивная функция вывода дерева на экран.

**Определить в классе конструкторы: по умолчанию, с параметрами, копирования. Определить деструктор.**

**РЕШЕНИЕ**

class Node

{

private:

char name[10];

Node \*left;

Node \*right;

bool isnull; // Нулевой объект

void Init();

public:

Node();// Пустой конструктор

Node(const char \*name);//Лаба 3

Node(const Node &node);//Лаба 3, конструктор копирования

void AddNode(const char \*name);

void DelTree();

void Print();

bool isNull();

~Node();//Лаба 3, деструктор

};

// Лаба 3

Node::Node(){

Init();

}

// Лаба 3

Node::Node(const char \*name){

Init();

int i = 0;

for (i = 0; i < 10 && name[i]; i++) this->name[i] = name[i];

if (i < 10) this->name[i] = 0;

isnull = false;

}

// Лаба 3, конструктор копирования

Node::Node(const Node &node){

if (this->left) this->left->DelTree();

if (this->right) this->right->DelTree();

memcpy(this, &node, sizeof(Node));

}

bool Node::isNull(){ return this->isnull; }

// Лаба 2

void Node::Init(){

left = right = NULL;

isnull = true;

memset(name, 0, 10);

}

// Лаба 2

void Node::AddNode(const char \*name){

static char buffer[12] = { 0 };

int i = 0;

// На случай, если длина строки равна 10

for (i = 0; i < 10 && this->name[i]; i++) buffer[i] = this->name[i];

buffer[i] = 0;

std::string s1(this->name), s2(name);

// Если вставляемая строка меньше текущей

if (s1 > s2){

if (!this->left){

this->left = new Node(name);

}

// Идти в левое поддерево

else{ this->left->AddNode(name); }

}

// Если вставляемая строка больше текущей

if (s1 < s2){

if (!this->right){

this->right = new Node(name);

}

// Идти в правое поддерево

else{ this->right->AddNode(name); }

}

}

// Лаба 2

void Node::DelTree(){

if (this->left) this->left->DelTree();

this->left = NULL;

if (this->right) this->right->DelTree();

this->right = NULL;

void \*data = this;

delete data;

}

// Лаба 2

void Node::Print(){

static char buffer[12] = { 0 };

int i = 0;

if (this->isnull){

std::cout << "null" << std::endl; return; }

// Вывести имя

for (i = 0; i < 10 && this->name[i]; i++) buffer[i] = this->name[i];

buffer[i] = 0;

std::cout << buffer << std::endl;

// Идти в левое поддерево, если оно есть

if (this->left) this->left->Print();

// Идти в правое поддерево, если оно есть

if (this->right) this->right->Print();

}

// Лаба 3, деструктор

Node::~Node(){

if (this->left) this->left->DelTree();

if (this->right) this->right->DelTree();

memset(this, 0, sizeof(Node));

}

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

std::cout << "Лабораторные 2 и 3" << std::endl;

std::cout << "Состояние дерева до вставки" << std::endl;

Node node("lab2");

node.Print();

node.AddNode("ty54rrtr");

node.AddNode("w33w0odr03w");

node.AddNode("3w032i3");

node.AddNode("node name");

std::cout << "Состояние дерева после вставки" << std::endl;

node.Print();

}

**4. ИНДИВИДУАЛЬНЫЕ ЗАДАНИЯ**

        Для всех вариантов необходимо выполнить следующее:

* определить классы их свойства и методы в соответствии с вариантом задания;
* в функции main() реализовать демонстрацию работы операций над объектами созданных классов;

**ЗАМЕЧАНИЯ:**

* Весь ввод/вывод должен осуществляться внутри функции main() если только это не оговорено для отдельных функций.
* Передача в функции объектов типа класс должна осуществляться через указатель либо ссылку на объект.
* Все другие нестатические данные должны передаваться функциям через параметры.
* Все нестатические элементы класса должны иметь закрытый атрибут доступа. Доступ извне к таким элементам должен осуществляться через функции-элементы класса.

**4)** Определите в классе **Node** следующие операции:

Node& operator=(const Node&);//копирование узла

Node& operator<<(const char\*);//добавить узел влево

Node& operator>(const char\*);//добавить узел вправо

Node& operator[](const char\*);//вернуть узел по его имени

friend ostream& operator<<(ostream& os, const Node&);//вывод

**РЕШЕНИЕ**

class Node

{

private:

char name[10];

Node \*left;

Node \*right;

bool isnull; // Нулевой объект

void Init();

public:

Node();// Пустой конструктор

Node(const char \*name);//Лаба 3

Node(const Node &node);//Лаба 3, конструктор копирования

void AddNode(const char \*name);

void DelTree();

void Print();

bool isNull();

~Node();//Лаба 3, деструктор

// Лаба 4

Node& getByName(const char\* name);

Node& operator=(const Node& another);

Node& operator<<(const char\* name);

Node& operator>(const char\* name);

Node& operator[](const char\* name);

friend std::ostream& operator<<(std::ostream& os, const Node&);

};

Node& Node::operator=(const Node& another){

if (this->left) this->left->DelTree();

if (this->right) this->right->DelTree();

memcpy(this, &another, sizeof(Node));

return \*this;

}

//Лаба 4, оператор вставки в левое поддерево

Node& Node::operator<<(const char\* name){

if (this->left) this->left->AddNode(name);

else{

this->left = new Node(name);

}

return \*this;

}

//Лаба 4, оператор вставки в правое поддерево

Node& Node::operator>(const char\* name){

if (this->right) this->right->AddNode(name);

else{

this->right = new Node(name);

}

return \*this;

}

//Лаба 4, взятие узла по имени

Node& Node::getByName(const char\* name){

static char buffer[12] = { 0 };

int i = 0;

static Node null;

for (i = 0; i < 10 && this->name[i]; i++) buffer[i] = this->name[i];

buffer[i] = 0;

std::string s1(buffer), s2(name);

if (s1 == s2) return \*this;// Вернуть текущий узел

// Искать в левом поддереве

if (s1>s2 && this->left) return this->left->getByName(name);

// Искать в правом поддереве

if (s1<s2 && this->right) return this->right->getByName(name);

return null;

}

Node& Node::operator[](const char \*name){

return this->getByName(name);

}

//Лаба 4, оператор вывода дерева

std::ostream& operator<<(std::ostream& os, const Node& node){

// Буфер для формирования Си строки

static char buffer[12] = { 0 };

int i = 0;

for (i = 0; i < 10 && node.name[i]; i++) buffer[i] = node.name[i];

buffer[i] = 0;

os << buffer<<std::endl;

//Распечатка левого поддерева

if (node.left) os << \*node.left;

//Распечатка правого поддерева

if (node.right) os << \*node.right;

return os;

}

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

// Лабораторная 4

std::cout << "Лабораторная 4" << std::endl;

std::cout << "node[node name]" << std::endl;

Node node1;

node1 = node["node name"];

node1.Print();

std::cout << "node[node]" << std::endl;

node1 = node["node"];

node1.Print();

std::cout << "Состояние дерева до вставки влево" << std::endl;

node.Print();

std::cout << "Состояние дерева после вставки влево" << std::endl;

node << "pogoda";

node.Print();

std::cout << "Состояние дерева после вставки вправо" << std::endl;

node > "snow";

node.Print();

std::cout << "Распечатка дерева через поток ostream" << std::endl;

std::cout << node << std::endl;

}