Учреждение образования

„Белорусский государственный университет информатики и радиоэлектроники”

Кафедра «Вычислительных методов и программирования»

ОТЧЕТ

По лабораторным работам №8-9

«Древовидные структуры данных. Вычисление алгебраических выражений (Польская запись)»

Выполнила:

Студентка АСОИ

Группы №820605

ФИО

Вариант № 9

Проверил:

ассистент кафедры ВМИП

Беспалов С.А.

Минск 2019

1. Индивидуальное задание:

В. *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.*

В9. Удалить все узлы дерева, имеющие значение ключа, равное 25.

Текст программы:

```
#include "pch.h"
#include <iostream>
using namespace std;

struct tree
{
	int inf;
	tree *left;
	tree *right;
};

void sort(int*, int);
tree *AddBalancedTree(int, int, int*);
void DirectBypass(tree*);
void InverseBypass(tree*);
void SymmBypass(tree*);
tree* DelNode(tree*, int);
void Symm(tree*, int);

bool b = 1;

int main()
{
	int n, del;
	cout << "Input number of elements: "; cin >> n;
	int* array = new int[n];
	cout << "Input elements: " << endl;
	for (int i = 0; i < n; i++)
		cin >> array[i];
	sort(array, n);
	tree *root = AddBalancedTree(0, n - 1, array);
	cout << "Direct bypass: "; DirectBypass(root);
	cout << endl << "Inverse bypass: "; InverseBypass(root);
	cout << endl << "Symmetric bypass: "; SymmBypass(root);
	cout << endl; Symm(root,0);
	cout << endl << "Input information to remove: "; cin >> del;
	root = DelNode(root, del);
	cout << endl << "Direct bypass: "; DirectBypass(root);
	cout << endl << "Inverse bypass: "; InverseBypass(root);
	cout << endl << "Symmetric bypass: "; SymmBypass(root);
	cout << endl; Symm(root, 0);	cout << endl;
	delete root;
	delete[] array;
	return 1;
}

void sort(int* array, int num)
{
	int j, extra;
	for (int i = 1; i < num; i++)
	{
```

```
                extra = array[i];
                for (j = i - 1; j >= 0 && extra < array[j]; j--) array[j + 1] = array[j];
                array[j + 1] = extra;
        }
}

tree *AddBalancedTree(int L, int R, int* array)
{
        tree *workaddress;
        int middle;
        if (L > R) return NULL;
        middle = (L + R) / 2;
        workaddress = new tree;
        workaddress->inf = array[middle];
        workaddress->left = AddBalancedTree(L, middle - 1, array);
        workaddress->right = AddBalancedTree(middle + 1, R, array);
        return workaddress;
}

void DirectBypass(tree* node)
{
        if (node == NULL) return;
        cout << (node->inf) << " ";
        DirectBypass(node->left);
        DirectBypass(node->right);
}

void InverseBypass(tree* node)
{
        if (node == NULL) return;
        InverseBypass(node->left);
        InverseBypass(node->right);
        cout << (node->inf) << " ";
}

void SymmBypass(tree* node)
{
        if (node == NULL) return;
        SymmBypass(node->left);
        cout << (node->inf) << " ";
        SymmBypass(node->right);
}

tree* DelNode(tree* root, int inf)
{
        tree *extra = root, *pre = root, *substitute, *presubstitute;
        //Поиск удаляемого узла

        while ((extra != NULL) && (extra->inf != inf))
        {
                pre = extra;
                if (inf < extra->inf)extra = extra->left;
                else extra = extra->right;
        }

        if (extra == NULL) return root; // Если узел не найден

        // Если узел не имеет дочерей
        if ((extra->left == NULL) && (extra->right == NULL))
        {
                if (extra == pre) // Если это был последний элемент
                {
                        delete (extra);
                        return NULL;
                }
```

```
        if (pre->left == extra) // Если удаляемый узел слева
                pre->left = NULL;
        else
                pre->right = NULL;
        delete(extra);
        return DelNode(root, inf);
}

// Если узел имеет дочь только справа
if (extra->left == NULL)
{
        if (extra == pre) // Если удаляется корень
        {
                extra = extra->right;
                delete(pre);
                return DelNode(extra, inf);
        }

        if (pre->left == extra) // Если удаляемый узел слева
                pre->left = extra->right;
        else                    // Если справа
                pre->right = extra->right;
        delete(extra);
        return DelNode(root, inf);
}

// Если узел имеет дочь только слева
if (extra->right == NULL)
{
        if (extra == pre) // Если удаляется корень
        {
                extra = extra->left;
                delete (pre);
                return DelNode(extra, inf);
        }

        if (pre->left==extra) // Если удаляемый узел слева
                pre->left = extra->left;
        else                    // Если справа
                pre->right = extra->left;
        delete(extra);
        return DelNode(root, inf);
}

//Если узел имеет двух дочерей
presubstitute = substitute = extra->left;
if (substitute->right == NULL) // Если substitute - максимальный из левых
substitute->right = extra->right;
else
{
        while (substitute->right != NULL)
        {
                presubstitute = substitute;
                substitute = substitute->right;
        }
        presubstitute->right = substitute->left;
        substitute->left = extra->left;
        substitute->right = extra->right;
}

if (extra == pre) // Если удаляется корень
{
        delete (extra);
        return DelNode(substitute, inf);
}
```

```
        if (pre->left == extra) // Если удаляемый узел слева
                pre->left = substitute;
        else                    // Если справа
                pre->right = substitute;
        delete(extra);
        return DelNode(root, inf);
}

void Symm(tree* root, int level)
{
        if (root == NULL)return;
        Symm(root->right, level + 1);
        for (int i = 0; i < level; i++)cout << "    ";
        cout << root->inf << endl;
        Symm(root->left, level + 1);
}
```

Результат :

```
Input number of elements: 10
Input elements:
1
25
90
0
87
25
46
3
25
8
Direct bypass: 25 1 0 3 8 46 25 25 87 90
Inverse bypass: 0 8 3 1 25 25 90 87 46 25
Symmetric bypass: 0 1 3 8 25 25 25 46 87 90
            90
        87
    46
            25
        25
25
            8
        3
    1
        0

Input information to remove: 25

Direct bypass: 8 1 0 3 46 87 90
Inverse bypass: 0 3 1 90 87 46 8
Symmetric bypass: 0 1 3 8 46 87 90
            90
        87
    46
8
        3
    1
        0
```

2. Индивидуальное задание:

B. *Ввести заданное арифметическое выражение и необходимые данные. Преобразовать запись арифметического выражения в форму обратной польской записи (для обозначения операции возведения в степень использовать знак ^). Вычислить арифметическое выражение. Результат вывести на экран. Задание выбрать в соответствии с номером варианта.*

B9. $x^w - y^w + \frac{a+y}{a-x}$.

Текст программы:

```
#include "pch.h"
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

using namespace std;

struct stack
{
	double inf;
	stack*a;
};

stack *AddStack(stack *sp, double inf)
{
	stack *spt = new stack;
	spt->inf = inf;
	spt->a = sp;
	return spt;
}

stack* ReadDelStack(stack*sp, double &inf)
{
	stack *spt = sp;
	inf = sp->inf;
	sp = sp->a;
	delete spt;
	return sp;
}


int priority(char sym) // Вычисление приоритета операций
{
	switch (sym)
	{
	case '(':   case ')':  return 0;
	case '+':   case '-':  return 1;
	case '*':   case '/':  return 2;
	case '^':              return 3;
	default:               return -1;
	}
}

void RevPolNot(char *strin, char *strout)
{
	stack *sp = NULL;
```

```
	int n = 0;
	char ch;
	double inf;

	for (unsigned int i = 0; i < strlen(strin); i++)
	{
		ch = strin[i];
		// Если это операнд
		if (ch >= 'A' && ch!='^') { strout[n++] = ch; continue; }
		// Если стек пуст или это открывающая скобка
		if (sp == NULL || ch == '(') { sp = AddStack(sp, ch); continue; }
		// Если найдена закрывающая скобка
		if (ch == ')')
		{
			while (sp->inf != '(')
			{
				sp = ReadDelStack(sp, inf);
				strout[n++] = (char)inf;
			}
			sp = ReadDelStack(sp, inf); // Удаление открывающей скобки
			continue;
		}
		// Если операция
		int pr = priority(ch);
		while (sp != NULL && priority((char)sp->inf) >= pr)
		{
			sp = ReadDelStack(sp, inf);
			strout[n++] = (char)inf;
		}
		sp = AddStack(sp, ch);
	}

	while (sp != NULL)
	{
		sp = ReadDelStack(sp, inf);
		strout[n++] = (char)inf;
	}
	strout[n++] = '\0';
}

double Calculation(char *expr, double *values)
{
	stack* sp = NULL;
	char symbol;
	double inf, inf1, inf2;
	for (unsigned int i = 0; i < strlen(expr); i++)
	{
		symbol = expr[i];
		// Если найден операнд
		if (symbol >= 'A' && symbol!= '^') { sp = AddStack(sp, values[int(symbol)-
65]); continue; }
		// Если найден знак операции
		sp = ReadDelStack(sp, inf2);
		sp = ReadDelStack(sp, inf1);
		switch (symbol)
		{
		case'+': sp = AddStack(sp, inf1 + inf2); break;
		case'-': sp = AddStack(sp, inf1 - inf2); break;
		case '*': sp = AddStack(sp, inf1*inf2); break;
		case '/': sp = AddStack(sp, inf1 / inf2); break;
		case '^': sp = AddStack(sp, pow(inf1, inf2)); break;
		}
	}
	sp = ReadDelStack(sp, inf);
	delete sp;
```

```
	return inf;
}

double* Values(char* expr)
{
	double* values = new double[58];
	for (int i = 0; i < 58; i++)
	{
		values[i] = INT_MAX;
	}
	for (int i = 0; i < strlen(expr); i++)
	{
		if ((expr[i] >= 'A' && expr[i] <= 'Z') || (expr[i] >= 'a' && expr[i] <=
'z'))
		{
			if (values[int(expr[i]) - 65] == INT_MAX)
			{
				cout << "Enter value of " << expr[i] << " : ";
				cin >> values[int(expr[i]) - 65];
			}
		}
	}
	return values;
}


int main()
{
	double* letters = new double[122];
	char expr[100], exprRPN[100];
	cout << "Input expression" << endl;
	cin >> expr;
	RevPolNot(expr, exprRPN);
	cout << endl << "Reverse Polish notation of your expression: " << exprRPN << endl
<< endl;
	letters = Values(exprRPN);
	double res = Calculation(exprRPN, letters);
	cout << endl << "Result = " << res << endl;
	delete[] letters;
	return 0;
}
```

Результат :

```
Input expression
x^w-y^w+(a+y)/(a-x)

Reverse Polish notation of your expression: xw^yw^-ay+ax-/+

Enter value of x : 2
Enter value of w : 4
Enter value of y : 3
Enter value of a : 1

Result = -69
```