**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

**отчет**

**по лабораторной работе №4**

**по дисциплине «Алгоритмы и структуры данных»**

**Тема: Бинарные деревья**

|  |  |  |
| --- | --- | --- |
| Студент гр. 6383 |  | Никитин К.В. |
| Преподаватель |  | Шолохова О.М. |

Санкт-Петербург

2017

**Задание (вариант №9 a) б) д)).**

Формулу вида

< формула > ::= < терминал > | ( < формула > < знак > < формула > )

< знак > ::= + | - | \*

< терминал > ::= 0 | 1 | ... | 9 | a | b | ... | z

можно представить в виде бинарного дерева («дерева-формулы») с элементами типа char согласно следующим правилам:

- формула из одного терминала представляется деревом из одной вершины с этим терминалом;

- формула вида (f1 s f2) представляется деревом, в котором корень - это знак s, а левое и правое поддеревья - соответствующие представления формул f1 и f2. Например, формула (5 \* (a + 3)) представляется деревом-формулой, показанной на рисунке ниже.

\*

5

*a*

+

3

Требуется:

а) для заданной формулы f построить дерево-формулу t;

б) для заданного дерева-формулы t напечатать соответствующую формулу f;

д) если в дереве-формуле t терминалами являются только цифры, то вычислить (как целое число) значение дерева-формулы t;

Исходные данные: программа поддерживает входные данные двух видов. Выбор вида входных осуществляется пользователем при выводе соответствующего запроса на экран.

Виды входных данных:

1. Формула вида

< формула > ::= < терминал > | ( < формула > < знак > < формула > )

< знак > ::= + | - | \*

< терминал > ::= 0 | 1 | ... | 9 | a | b | ... | z

1. Дерево-формула вида

<дерево> ::= ( <терминал> | <знак> <дерево> <дерево> )

< знак > ::= + | - | \*

< терминал > ::= 0 | 1 | ... | 9 | a | b | ... | z

Результирующие данные: Вывод на экран и в файл данные, соответствующие выбору пользователя (дерево-формула, формула, значение формулы, дерево-производная формулы по заданной переменной).

**Спецификация программы.**

Ограничения на входные данные: Выражение, принимающееся на входе должно быть записано синтаксически правильно.

Место и форма представления входных данных: входные данные располагаются в файле input.txt или вводятся с клавиатуры (при выполнении программы пользователю предоставляется выбор).

Выходные данные: Программа выводит на экран и в файл соответствующие запросу пользователя данные.

*Реализация*.

1. Функция, выводящая на экран и в файл сообщение об ошибке входных данных и производящая выход из программы.

void ErrorMessege(ofstream& f1){

cout << "Error! Uncorrect input. The programm will be interructed.\n";

system("pause");

exit(1);

}

ofstream& f1 – ссылка на переменную, содержащую данные выходного файла

1. Функция, проверяющая на корректность введенные в меню пользователем данные

void Error(int& w){

while ((cin.fail()) || ((w != 1) && (w != 2))){

cout << "Error. Enter another value.\n";

cin.clear();

cin.sync();

cin >> w;

}

}

int& w – ссылка на переменную хранящую введенное пользователем значение

1. Функция, выводящая сообщение об ошибке и прерывающая работу программы при некорректности входных данных

void ErrorMessege(ofstream& f1){

cout << "Error! Uncorrect input. The programm will be interrupted.\n";

f1 << "Error! Uncorrect input. The programm will be interrupted.\n";

system("pause");

exit(1);

}

ofstream& f1 – ссылка на переменную, содержащую данные выходного файла

1. Функция считывания формулы с клавиатуры

void read\_form(Formula\*& s, char& x, ofstream& f1){ // рекурсивная функция считывания формулы с клавиатуры

cin.get(x);

if ((x >= '0') && (x <= '9')){

s->tag = 1;

s->form.term.num = int(x) - 48;

}

else{

if ((x >= 'a') && (x <= 'z')){

s->tag = 2;

s->form.term.let = x;

}

else{

if (x != '(') ErrorMessege(f1);

else{

s->tag = 0;

s->form.form1.f1 = new Formula;

read\_form(s->form.form1.f1, x,f1);

cin.get(x);

if ((x != '\*') && (x != '+') && (x != '-')) ErrorMessege(f1);

else{

s->form.form1.znak = x;

}

s->form.form1.f2 = new Formula;

read\_form(s->form.form1.f2, x, f1);

cin.get(x);

if (x != ')') ErrorMessege(f1);

}

}

}

}

1. Функция считывания дерева-формулы с клавиатуры

void read\_three(Formula\*& s, char& x, ofstream& f1){ // рекурсивная функция считывания дерева-формулы с клавиатуры

cin.get(x);

if (x != '(') ErrorMessege(f1);

else{

cin.get(x);

if ((x == '\*') || (x == '+') || (x == '-')){

s->tag = 0;

s->form.form1.znak = x;

s->form.form1.f1 = new Formula;

s->form.form1.f2 = new Formula;

read\_three(s->form.form1.f1, x, f1);

read\_three(s->form.form1.f2, x, f1);

}

else{

if ((x >= '0') && (x <= '9')){

s->tag = 1;

s->form.term.num = int(x) - 48;

}

else{

if ((x >= 'a') && (x <= 'z')){

s->tag = 2;

s->form.term.let = x;

}

else ErrorMessege(f1);

}

}

cin.get(x);

if (x != ')') ErrorMessege(f1);

}

}

1. Функция печати формулы

void print(Formula\*& s, ofstream& f1){

if (s->tag == 1){

cout << s->form.term.num;

f1 << s->form.term.num;

}

if (s->tag == 2){

cout << s->form.term.let;

f1 << s->form.term.let;

}

if (!s->tag){

cout << "(";

f1 << "(";

print(s->form.form1.f1, f1);

cout << s->form.form1.znak;

f1 << s->form.form1.znak;

print(s->form.form1.f2, f1);

cout << ")";

f1 << ")";

}

}

1. Функция расчета результата формулы

void calculate(Formula\*& s, int& res, bool& k, ofstream& f1){

if (k){

if (s->tag == 2) k = 0;

else{

if (s->tag == 1){

res = s->form.term.num;

}

else{

int res1, res2;

calculate(s->form.form1.f1, res1, k, f1);

calculate(s->form.form1.f2, res2, k, f1);

if (s->form.form1.znak == '+') res = res1 + res2;

if (s->form.form1.znak == '-') res = res1 - res2;

if (s->form.form1.znak == '\*') res = res1 \* res2;

if (k){

cout << "The value of ";

f1 << "The value of ";

print(s, f1);

cout << " is " << res << endl;

f1 << " is " << res << endl;

}

}

}

}

}

1. Функция, изображающая графически дерево-формулу

void build(Formula\*& s, ofstream& f1, int& depth, bool close[]){

depth++;

if (s->tag == 1){

cout << s->form.term.num << endl;

f1 << s->form.term.num << endl;

}

if (s->tag == 2){

cout << s->form.term.let << endl;

f1 << s->form.term.let << endl;

}

if (!s->tag){

cout << s->form.form1.znak << endl;

f1 << s->form.form1.znak << endl;

for (int j = 1; j <= depth; j++){

if (!close[j]){

cout << "| ";

f1 << "| ";

}

else{

cout << " ";

f1 << " ";

}

}

cout << endl;

f1 << endl;

for (int j = 1; j <= depth - 1; j++){

if (!close[j]){

cout << "| ";

f1 << "| ";

}

else{

cout << " ";

f1 << " ";

}

}

cout << "|--";

f1 << "|--";

build(s->form.form1.f1, f1, depth, close);

depth--;

for (int j = 1; j <= depth; j++){

if (!close[j]){

cout << "| ";

f1 << "| ";

}

else{

cout << " ";

f1 << " ";

}

}

cout << endl;

f1 << endl;

for (int j = 1; j <= depth - 1; j++){

if (!close[j]){

cout << "| ";

f1 << "| ";

}

else{

cout << " ";

f1 << " ";

}

}

cout << "|--";

f1 << "|--";

close[depth] = 1;

build(s->form.form1.f2, f1, depth, close);

depth--;

close[depth] = 0;

}

}

**Тестирование.**

|  |  |  |
| --- | --- | --- |
| № | Входные данные | Выходные данные |
| 1. |  | Error! Uncorrect input. The programm will be interrupted. |
| 2. | (8\*9)) | Error! Uncorrect input. The programm will be interrupted. |
| 3. | (((3+2)\*(8\*9))+((8\*9)-(3\*(7-9))))  1 // вывод дерева | (((3+2)\*(8\*9))+((8\*9)-(3\*(7-9))))  +  |  |--\*  | |  | |--+  | | |  | | |--3  | | |  | | |--2  | |  | |--\*  | |  | |--8  | |  | |--9  |  |---  |  |--\*  | |  | |--8  | |  | |--9  |  |--\*  |  |--3  |  |---  |  |--7  |  |--9 |
| 4. | (((3+2)\*(8\*9))+((8\*9)-(3\*(7-9))))  3// Посчитать значение | (3+2) = 5  (8\*9) = 72  ((3+2)\*(8\*9)) = 360  (8\*9) = 72  (7-9) = -2  (3\*(7-9)) = -6  ((8\*9)-(3\*(7-9))) = 78  (((3+2)\*(8\*9))+((8\*9)-(3\*(7-9)))) = 438 |
| 5. | (((a+2)\*(a\*a))+((8\*c)-(b\*(a-9))))  3// Посчитать значение | (((a+2)\*(a\*a))+((8\*c)-(b\*(a-9))))  Calculation can not be performed. |
| 6. | (+(-(0)(9))(5))  3 // Рассчитать значение формулы | (0-9) = -9  ((0-9)+5) = -4 |
| 7. | (+(-(0)(9))(\*(+(a)(c))(\*(a)(3))))  2// Напечатать формулу | ((0-9)+((a+c)\*(a\*3))) |

**Вывод.**

В результате выполнения данной работы были освоены навыки использования бинарных деревьев . Закреплены навыки использования динамической памяти.

**Приложение А. Исходный код.**

**Source.cpp**

#include <iostream>

#include <fstream>

#include <cstdlib>

#include "head.h"

#include <windows.h>

using namespace std;

void main() {

int w = 1;

char x;

Formula\* head = new Formula;

cout << "Please, select:\n1. Enter the three\n2. Enter the formula\n";

cin >> w;

Error(w);

cin.get(x);

if (w == 1) read\_three(head, x); // вызов функции считывания дерева-формулы с клавиатуры

if (w == 2) read\_form(head, x); // вызов функции считывания формулы с клавиатуры

cin.get(x);

if (x != '\n') ErrorMessege(); // если не все данные были считаны, то вывод об ошибке и прерывание программы

system("cls");

while (w != 4) {// выбор пользователем последующей работы программы

system("cls");

cout << "Please, select:\n1. Print the three\n2. Print the formula\n3. Calculate the value\n4. Exit\n";

cin >> w;

switch (w) {

case 1: // печать дерева-формулы

int depth; // переменная для определения текущей глубины формулы

depth = 0;

bool close[50]; // массив переменных для определения "закрытости" дерева с соответствующей глубиной, то есть пограмма работает со вторым его плечом.

for (int k = 1; k <= 49; k++) close[k] = 0;

print(head); // вызов функции печати формулы

cout << endl;

build(head, depth, close); // вызов функции печати дерева-формулы

break;

case 2: // печать формулы

print(head); // вызов функции печати формулы

cout << endl;

break;

case 3: // расчет результата формулы

bool k;

k = 1; // переменная, определяющая наличие терминалов-букв

int res; // переменная для записи в нее результата

calculate(head, res, k); // вызов функции расчета результата

print(head); // вызов функции печати формулы

if (!k) { // печать о невозможности расчета результата из-за наличия в формуле не только цифр

cout << "\nCalcutation can not be performed.\n";

}

else {// печать результата

cout << " = " << res << endl;

}

break;

case 4:

w = 4;

break;

default: // некорректный ввод

cout << "Uncorrect input. Try again.\n";

break;

}

system("pause");

}

}

**func.cpp**

#include "head.h"

#include <iostream>

#include <fstream>

using namespace std;

void Error(int& w) {

while ((cin.fail()) || ((w != 1) && (w != 2))) {

cout << "Error. Enter another value.\n";

cin.clear();

cin.sync();

cin >> w;

}

}

void ErrorMessege() {

cout << "Error! Uncorrect input. The programm will be interrupted.\n";

system("pause");

exit(1);

}

void read\_form(Formula\*& s, char& x) { // рекурсивная функция считывания формулы с клавиатуры

cin.get(x);

if ((x >= '0') && (x <= '9')) {

s->tag = 1;

s->form.term.num = int(x) - 48;

}

else {

if ((x >= 'a') && (x <= 'z')) {

s->tag = 2;

s->form.term.let = x;

}

else {

if (x != '(') ErrorMessege();

else {

s->tag = 0;

s->form.form1.f1 = new Formula;

read\_form(s->form.form1.f1, x);

cin.get(x);

if ((x != '\*') && (x != '+') && (x != '-')) ErrorMessege();

else {

s->form.form1.znak = x;

}

s->form.form1.f2 = new Formula;

read\_form(s->form.form1.f2, x);

cin.get(x);

if (x != ')') ErrorMessege();

}

}

}

}

void read\_three(Formula\*& s, char& x) { // рекурсивная функция считывания дерева-формулы с клавиатуры

cin.get(x);

if (x != '(') ErrorMessege();

else {

cin.get(x);

if ((x == '\*') || (x == '+') || (x == '-')) {

s->tag = 0;

s->form.form1.znak = x;

s->form.form1.f1 = new Formula;

s->form.form1.f2 = new Formula;

read\_three(s->form.form1.f1, x);

read\_three(s->form.form1.f2, x);

}

else {

if ((x >= '0') && (x <= '9')) {

s->tag = 1;

s->form.term.num = int(x) - 48;

}

else {

if ((x >= 'a') && (x <= 'z')) {

s->tag = 2;

s->form.term.let = x;

}

else ErrorMessege();

}

}

cin.get(x);

if (x != ')') ErrorMessege();

}

}

void print(Formula\*& s) {

if (s->tag == 1) {

cout << s->form.term.num;

}

if (s->tag == 2) {

cout << s->form.term.let;

}

if (!s->tag) {

cout << "(";

print(s->form.form1.f1);

cout << s->form.form1.znak;

print(s->form.form1.f2);

cout << ")";

}

}

void calculate(Formula\*& s, int& res, bool& k) {

if (k) {

if (s->tag == 2) k = 0;

else {

if (s->tag == 1) {

res = s->form.term.num;

}

else {

int res1, res2;

calculate(s->form.form1.f1, res1, k);

calculate(s->form.form1.f2, res2, k);

if (s->form.form1.znak == '+') res = res1 + res2;

if (s->form.form1.znak == '-') res = res1 - res2;

if (s->form.form1.znak == '\*') res = res1 \* res2;

if (k) {

cout << "The value of ";

print(s);

cout << " is " << res << endl;

}

}

}

}

}

void build(Formula\*& s, int& depth, bool close[]) {

depth++;

if (s->tag == 1) {

cout << s->form.term.num << endl;

}

if (s->tag == 2) {

cout << s->form.term.let << endl;

}

if (!s->tag) {

cout << s->form.form1.znak << endl;

for (int j = 1; j <= depth; j++) {

if (!close[j]) {

cout << "| ";

}

else {

cout << " ";

}

}

cout << endl;

for (int j = 1; j <= depth - 1; j++) {

if (!close[j]) {

cout << "| ";

}

else {

cout << " ";

}

}

cout << "|--";

build(s->form.form1.f1, depth, close);

depth--;

for (int j = 1; j <= depth; j++) {

if (!close[j]) {

cout << "| ";

}

else {

cout << " ";

}

}

cout << endl;

for (int j = 1; j <= depth - 1; j++) {

if (!close[j]) {

cout << "| ";

}

else {

cout << " ";

}

}

cout << "|--";

close[depth] = 1;

build(s->form.form1.f2, depth, close);

depth--;

close[depth] = 0;

}

}

**head.h**

#pragma once

#include <iomanip>

using namespace std;

struct Formula;

struct Formula1 {

Formula\* f1;

Formula\* f2;

char znak;

};

struct Formula {

int tag;

union {

union {

int num;

char let;

}term;

Formula1 form1;

}form;

};

void read\_three(Formula\*& s, char& x);

void read\_form(Formula\*& s, char& x);

void build(Formula\*& s, int& depth, bool close[]);

void print(Formula\*& s);

void calculate(Formula\*& s, int& res, bool& k);

void ErrorMessege();

void Error(int& w);