Содержание

Практическая работа №9

Вариант №3 – Префиксный калькулятор

Автор: Николаев-Аксенов И. С.

Группа: ИКБО-20-19

Код программы:

```
1.  package Calculator;
2.  import java.util.Scanner;
3.  import java.util.Stack;
4.
5.  public class Calculator {
6.      public static int evaluate(String mathLine) {
7.          char[] nums = mathLine.toCharArray();
8.
9.          Stack<Integer> values = new Stack<>();
10.         Stack<Character> ops = new Stack<>();
11.
12.         for (int i = 0; i < nums.length; i++) {
13.             if (nums[i] == ' ')
14.                 continue;
15.
16.             if (nums[i] >= '0' && nums[i] <= '9') {
17.                 StringBuffer sbuf = new StringBuffer();
18.                 while (i < nums.length && nums[i] >= '0' && nums[i] <= '9')
19.                     sbuf.append(nums[i++]);
20.                 values.push(Integer.parseInt(sbuf.toString()));
21.             }
22.
23.             else if (nums[i] == '(')
24.                 ops.push(nums[i]);
25.
26.             else if (nums[i] == ')') {
27.                 while (ops.peek() != '(')
28.                     values.push(resultOperation(ops.pop(), values.pop(), values.pop()));
29.                 ops.pop();
30.             }
31.
32.             else if (nums[i] == '+' || nums[i] == '-' || nums[i] == '*' || nums[i] ==
    '/') {
33.                 while (!ops.empty() && operationRangs(nums[i], ops.peek()))
34.                     values.push(resultOperation(ops.pop(), values.pop(), values.pop()));
35.
36.                 ops.push(nums[i]);
37.             }
38.         }
39.
40.         while (!ops.empty())
41.             values.push(resultOperation(ops.pop(), values.pop(), values.pop()));
42.
43.         return values.pop();
44.     }
45.
46.     public static boolean operationRangs(char op1, char op2) {
47.         if (op2 == '(' || op2 == ')')
48.             return false;
49.
50.         return (op1 != '*' && op1 != '/') || (op2 != '+' && op2 != '-');
51.     }
52.
53.     public static int resultOperation(char op, int b, int a) {
54.         switch (op) {
55.             case '+':
56.                 return a + b;
```

```
            case '-':
                return a - b;
            case '*':
                return a * b;
            case '/':
                if (b == 0)
                    throw new
                            UnsupportedOperationException("На нуль делить нельзя!");
                return a / b;
        }
        return 0;
    }

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("Введите арифметическое выражение через пробел: ");
        String mathLine = input.nextLine();
        System.out.print("Ответ: ");
        System.out.println(Calculator.evaluate(mathLine));
    }
}
```

Результат выполнения программы:

```
Введите арифметическое выражение через пробел:
10 + 2 * ( 5 + 5 )
Ответ: 30
```

Практическая работа №10

Бинарное дерево поиска. AVL дерево

Автор: Николаев-Аксенов И. С.

Группа: ИКБО-20-19

Код программы:

RedBlackTree.py:

```
import sys


class Node():
    def __init__(self, data):
        self.data = data
        self.parent = None
        self.left = None
        self.right = None
        self.color = 1


class RedBlackTree():
    def __init__(self):
        self.TNULL = Node(0)
        self.TNULL.color = 0
        self.TNULL.left = None
        self.TNULL.right = None
        self.root = self.TNULL

    def __pre_order_helper(self, node):
        if node != TNULL:
            sys.stdout.write(node.data + " ")
            self.__pre_order_helper(node.left)
            self.__pre_order_helper(node.right)

    def __in_order_helper(self, node):
        if node != TNULL:
            self.__in_order_helper(node.left)
            sys.stdout.write(node.data + " ")
            self.__in_order_helper(node.right)

    def __post_order_helper(self, node):
        if node != TNULL:
            self.__post_order_helper(node.left)
            self.__post_order_helper(node.right)
            sys.stdout.write(node.data + " ")

    def __search_tree_helper(self, node, key):
        if node == TNULL or key == node.data:
            return node

        if key < node.data:
            return self.__search_tree_helper(node.left, key)
        return self.__search_tree_helper(node.right, key)

    def __fix_delete(self, x):
        while x != self.root and x.color == 0:
            if x == x.parent.left:
                s = x.parent.right
                if s.color == 1:
                    s.color = 0
                    x.parent.color = 1
                    self.left_rotate(x.parent)
```

```
                    s = x.parent.right

                if s.left.color == 0 and s.right.color == 0:
                    s.color = 1
                    x = x.parent
                else:
                    if s.right.color == 0:
                        s.left.color = 0
                        s.color = 1
                        self.right_rotate(s)
                        s = x.parent.right

                    s.color = x.parent.color
                    x.parent.color = 0
                    s.right.color = 0
                    self.left_rotate(x.parent)
                    x = self.root
            else:
                s = x.parent.left
                if s.color == 1:
                    s.color = 0
                    x.parent.color = 1
                    self.right_rotate(x.parent)
                    s = x.parent.left

                if s.left.color == 0 and s.right.color == 0:
                    s.color = 1
                    x = x.parent
                else:
                    if s.left.color == 0:
                        s.right.color = 0
                        s.color = 1
                        self.left_rotate(s)
                        s = x.parent.left

                    s.color = x.parent.color
                    x.parent.color = 0
                    s.left.color = 0
                    self.right_rotate(x.parent)
                    x = self.root
        x.color = 0

    def __rb_transplant(self, u, v):
        if u.parent == None:
            self.root = v
        elif u == u.parent.left:
            u.parent.left = v
        else:
            u.parent.right = v
        v.parent = u.parent

    def __delete_node_helper(self, node, key):
        z = self.TNULL
        while node != self.TNULL:
            if node.data == key:
                z = node

            if node.data <= key:
                node = node.right
            else:
                node = node.left

        if z == self.TNULL:
            print("Данный ключ не найден на дереве")
            return

        y = z
        y_original_color = y.color
        if z.left == self.TNULL:
```

```
            x = z.right
            self.__rb_transplant(z, z.right)
        elif (z.right == self.TNULL):
            x = z.left
            self.__rb_transplant(z, z.left)
        else:
            y = self.minimum(z.right)
            y_original_color = y.color
            x = y.right
            if y.parent == z:
                x.parent = y
            else:
                self.__rb_transplant(y, y.right)
                y.right = z.right
                y.right.parent = y

            self.__rb_transplant(z, y)
            y.left = z.left
            y.left.parent = y
            y.color = z.color
        if y_original_color == 0:
            self.__fix_delete(x)

    def __fix_insert(self, k):
        while k.parent.color == 1:
            if k.parent == k.parent.parent.right:
                u = k.parent.parent.left  # uncle
                if u.color == 1:
                    u.color = 0
                    k.parent.color = 0
                    k.parent.parent.color = 1
                    k = k.parent.parent
                else:
                    if k == k.parent.left:
                        k = k.parent
                        self.right_rotate(k)
                    k.parent.color = 0
                    k.parent.parent.color = 1
                    self.left_rotate(k.parent.parent)
            else:
                u = k.parent.parent.right

                if u.color == 1:
                    u.color = 0
                    k.parent.color = 0
                    k.parent.parent.color = 1
                    k = k.parent.parent
                else:
                    if k == k.parent.right:
                        k = k.parent
                        self.left_rotate(k)
                    k.parent.color = 0
                    k.parent.parent.color = 1
                    self.right_rotate(k.parent.parent)
            if k == self.root:
                break
        self.root.color = 0

    def __print_helper(self, node, indent, last):
        if node != self.TNULL:
            sys.stdout.write(indent)
            if last:
                sys.stdout.write("R----")
                indent += "     "
            else:
                sys.stdout.write("L----")
                indent += "|    "

            s_color = "RED" if node.color == 1 else "BLACK"
```

```
193.             print(str(node.data) + "(" + s_color + ")")
194.             self.__print_helper(node.left, indent, False)
195.             self.__print_helper(node.right, indent, True)
196.
197.     def preorder(self):
198.         self.__pre_order_helper(self.root)
199.
200.     def inorder(self):
201.         self.__in_order_helper(self.root)
202.
203.     def postorder(self):
204.         self.__post_order_helper(self.root)
205.
206.     def searchTree(self, k):
207.         return self.__search_tree_helper(self.root, k)
208.
209.     def minimum(self, node):
210.         while node.left != self.TNULL:
211.             node = node.left
212.         return node
213.
214.     def maximum(self, node):
215.         while node.right != self.TNULL:
216.             node = node.right
217.         return node
218.
219.     def successor(self, x):
220.         if x.right != self.TNULL:
221.             return self.minimum(x.right)
222.         y = x.parent
223.         while y != self.TNULL and x == y.right:
224.             x = y
225.             y = y.parent
226.         return y
227.
228.     def predecessor(self,  x):
229.         if (x.left != self.TNULL):
230.             return self.maximum(x.left)
231.
232.         y = x.parent
233.         while y != self.TNULL and x == y.left:
234.             x = y
235.             y = y.parent
236.
237.         return y
238.
239.     def left_rotate(self, x):
240.         y = x.right
241.         x.right = y.left
242.         if y.left != self.TNULL:
243.             y.left.parent = x
244.
245.         y.parent = x.parent
246.         if x.parent == None:
247.             self.root = y
248.         elif x == x.parent.left:
249.             x.parent.left = y
250.         else:
251.             x.parent.right = y
252.         y.left = x
253.         x.parent = y
254.
255.     def right_rotate(self, x):
256.         y = x.left
257.         x.left = y.right
258.         if y.right != self.TNULL:
259.             y.right.parent = x
260.
261.         y.parent = x.parent
```

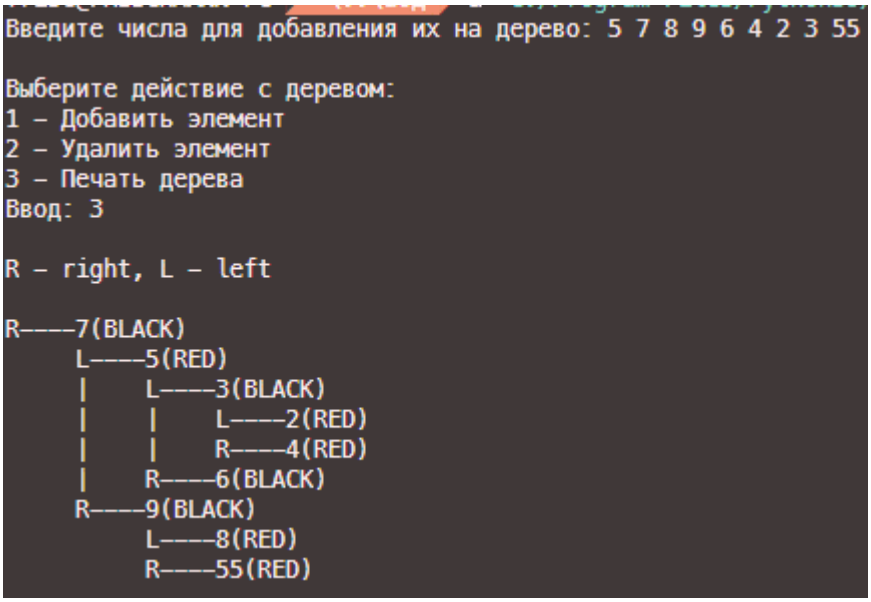
```
262.             if x.parent == None:
263.                 self.root = y
264.             elif x == x.parent.right:
265.                 x.parent.right = y
266.             else:
267.                 x.parent.left = y
268.             y.right = x
269.             x.parent = y
270.
271.         def insert(self, key):
272.             node = Node(key)
273.             node.parent = None
274.             node.data = key
275.             node.left = self.TNULL
276.             node.right = self.TNULL
277.             node.color = 1
278.
279.             y = None
280.             x = self.root
281.
282.             while x != self.TNULL:
283.                 y = x
284.                 if node.data < x.data:
285.                     x = x.left
286.                 else:
287.                     x = x.right
288.             node.parent = y
289.             if y == None:
290.                 self.root = node
291.             elif node.data < y.data:
292.                 y.left = node
293.             else:
294.                 y.right = node
295.
296.             if node.parent == None:
297.                 node.color = 0
298.                 return
299.
300.             if node.parent.parent == None:
301.                 return
302.
303.             self.__fix_insert(node)
304.
305.         def get_root(self):
306.             return self.root
307.
308.         def delete_node(self, data):
309.             self.__delete_node_helper(self.root, data)
310.
311.         def pretty_print(self):
312.             self.__print_helper(self.root, "", True)
```

startRBT.py:

```
1.  from RedBlackTree import RedBlackTree
2.  rbt = RedBlackTree()
3.
4.
5.  def menu():
6.      x = int(input("\nВыберите действие с деревом:\n1 - Добавить элемент\n2 - Удалить
    элемент\n3 - Печать дерева\nВвод: "))
7.      if (x == 1):
8.          rbt.insert(int(input("Введите число для добавления его на дерево: ")))
9.          print("Число успешно добавлено на дерево!\n")
10.         menu()
11.     elif (x == 2):
12.         rbt.delete_node(
```

```
            int(input("Введите число которое вы хотите удалить: ")))
        print("Число успешно удалено из дерева!\n")
        menu()
    elif (x == 3):
        print("\nR - right, L - left\n")
        rbt.pretty_print()
        menu()
    else:
        print("Действие не найдено! Повторите ввод.")
        menu()


def main():
    numbers = list(
        map(int, input("Введите числа для добавления их на дерево: ").split()))

    for i in numbers:
        rbt.insert(i)

    menu()


if __name__ == "__main__":
    main()
```

Результат выполнения программы:



Практическая работа №11

Вариант №7 – Цепное хеширование. Страховой полис: номер, компания, фамилия владельца.

Автор: Николаев-Аксенов И. С.

Группа: ИКБО-20-19

Код программы:

Insurance.java:

```java
public class Insurance {
    int number;
    String company;
    String surname;

    public Insurance(int number, String company, String surname) {
        this.number = number;
        this.company = company;
        this.surname = surname;
    }

    public int getNumber() {
        return number;
    }

    @Override
    public String toString() {
        return "Insurance {" +
                "number=" + number +
                ", company='" + company + '\'' +
                ", surname='" + surname + '\'' +
                '}'+"\n";
    }
}
```

HashTable.java:

```java
import java.util.*;

public class HashTable< E> {
    ArrayList<LinkedList<E>>table;
    int size;

    public HashTable(int size) {
        this.size = size;
        this.table = new ArrayList<>(this.size);

        for(int i=0;i < 10;i++){
            table.add(new LinkedList<E>());
        }
    }

    int hash(int value){
        return (int)value%size;
    }

    int hash(E n) {
        Insurance key = (Insurance) n;
        return key.number % size;
    }
```

```
24.
25.     void add(E b) {
26.         table.get(hash(b)).addLast(b);
27.         if (table.get(hash(b)).size() > 2) rehash();
28.     }
29.
30.     void rehash(){
31.         ArrayDeque<E> t =new ArrayDeque<>();
32.         for (int i = 0; i < size; ++i) {
33.             for (E el : table.get(i)) {
34.                 t.add(el);
35.             }
36.         }
37.         size = size * 2 + 1;
38.         table.clear();
39.         table = new ArrayList<>(size);
40.         for(int i=0;i<size;i++){
41.             table.add(new LinkedList<E>());
42.         }
43.         while (!t.isEmpty()) {
44.             add(t.getFirst());
45.             t.pop();
46.         }
47.     }
48.     void search(int value){
49.         for(E t:table.get(hash(value))){
50.             Insurance c= (Insurance) t;
51.             if(c.number == value){
52.                 System.out.println(c);
53.             }
54.         }
55.     }
56.
57.     void delete(int value){
58.         for(E t:table.get(hash(value))){
59.             Insurance c= (Insurance) t;
60.             if(c.number == value){
61.                 table.get(hash(value)).remove(t);
62.             }
63.         }
64.     }
65.     void print(){
66.         for(int i=0;i<size;i++){
67.             if(!table.get(i).isEmpty()) {
68.                 System.out.println(i+" : ");
69.                 for (int j = 0; j < table.get(i).size(); j++) {
70.                     System.out.println("\t"+table.get(i).get(j));
71.                 }
72.             }
73.         }
74.     }
75. }
```

StartInsurance.java:

```java
public class StartInsurance {
    public static void main(String[] args) {
        HashTable<Insurance> ht = new HashTable<>(10);
        ht.add(new Insurance(123,"yandex","ivanov"));
        ht.add(new Insurance(1005,"google","nikolaev"));
        ht.add(new Insurance(5,"amazon","axenov"));

        System.out.println("Вывод до рехеширования");
        ht.print();
        ht.add(new Insurance(2005,"netflix","ivanov"));
        ht.add(new Insurance(3005,"tesla","nikolaev"));

        System.out.println("Вывод после рехеширования");
        ht.print();
        System.out.println("\n");

        ht.search(1005);
        System.out.println("\n");

        ht.delete(1005);
        System.out.println("\n");

        System.out.println("Вывод после удаления элемента");
        ht.print();
    }
}
```

Результат выполнения программы:

```
Вывод до рехеширования
3 :
    Insurance {number=123, company='yandex', surname='ivanov'}

5 :
    Insurance {number=1005, company='google', surname='nikolaev'}

    Insurance {number=5, company='amazon', surname='axenov'}

Вывод после рехеширования
2 :
    Insurance {number=3005, company='tesla', surname='nikolaev'}

5 :
    Insurance {number=5, company='amazon', surname='axenov'}

10 :
    Insurance {number=2005, company='netflix', surname='ivanov'}

18 :
    Insurance {number=123, company='yandex', surname='ivanov'}

    Insurance {number=1005, company='google', surname='nikolaev'}



Insurance {number=1005, company='google', surname='nikolaev'}
```

```
Вывод после удаления элемента
2 :
    Insurance {number=3005, company='tesla', surname='nikolaev'}

5 :
    Insurance {number=5, company='amazon', surname='axenov'}

10 :
    Insurance {number=2005, company='netflix', surname='ivanov'}

18 :
    Insurance {number=123, company='yandex', surname='ivanov'}


Process finished with exit code 0
```

Практическая работа №12

Вариант №2 – Счет в банке.

Автор: Николаев-Аксенов И. С.

Группа: ИКБО-20-19

Код программы:

BankAccount.cpp:

```
#include <iostream>
#include <string>
#include <fstream>
#include <vector>
using namespace std;

struct bank_account
{
    int num;
    string name;
    string surname;
    string second_name;
    string address;
};
vector<bank_account>bk;

void print_data(bank_account a)
{
    cout << "Number : " << a.num << "\nName : " << a.name << "\nSurname : " << a.surname << "\nSecond Name : " << a.second_name << "\nAddress : " << a.address << endl;
    cout << "_________________\n";
}

void write_data_to_file()
{
    ofstream ou("bank.txt");

    if (ou.is_open())
    {
        for (int i = 0; i < bk.size(); i++)
        {
            bank_account temp = bk[i];
            ou << temp.num << " " << temp.name << " "<< temp.surname << " "<< temp.second_name << " "<< temp.address<< "\n";
        }
    }
    ou.close();
}

void write_data_to_binary()
{
    bank_account temp;
    ofstream out("binary.txt", ostream::binary);
    for (int i = 0; i <bk.size(); ++i)
        out.write((char*)&bk.at(i), sizeof(bank_account));

    out.close();
}

void read_data_from_binary_file()
{
    ifstream fin("binary.txt", istream::binary);
    bank_account temp;
    for (int i = 0; i < bk.size(); i++)
```

```
53.         fin.read((char*)&bk.at(i), sizeof(bank_account));
54.
55.     fin.close();
56. }
57.
58. void print_all_data()
59. {
60.     for (int i = 0; i < bk.size(); i++)
61.         print_data(bk[i]);
62. }
63.
64. void find()
65. {
66.     int a;
67.     cout << "Input number of account, which you want to find: ";
68.     cin >> a;
69.     cout << endl;
70.     for (int i = 0; i < bk.size(); i++)
71.     {
72.         if (bk[i].num == a)
73.             print_data(bk[i]);
74.     }
75. }
76.
77. void change()
78. {
79.     int a;
80.     string b;
81.     cout << "Input number of account, which you want to change : ";
82.     cin >> a;
83.     cout << endl;
84.
85.     cout << "Input new name : ";
86.     cin >> b;
87.     cout << endl;
88.
89.     read_data_from_binary_file();
90.     for (int i = 0; i < bk.size(); i++)
91.     {
92.         if (bk[i].num == a)
93.             bk[i].name = b;
94.     }
95.     write_data_to_binary();
96. }
97.
98. void del()
99. {
100.     int a;
101.     cout << "Input number of account, which you want to delete : ";
102.     cin >> a;
103.     cout << endl;
104.     read_data_from_binary_file();
105.
106.     for (int i = 0; i < bk.size(); i++){
107.         if (bk[i].num == a)
108.             bk.erase(bk.begin() + i);
109.     }
110.     write_data_to_binary();
111. }
112.
113. int main()
114. {
115.     bank_account first{1234567, "Ivan", "Ivanov", "Ivanov", "Perm"};
116.     bk.push_back(first);
117.
118.     bank_account second{7654321, "Alex", "Alexandrov", "Sergeevich", "Moscow"};
119.     bk.push_back(second);
120.
121.     write_data_to_file();
```

```
122.     print_all_data();
123.     cout << "_____\n";
124.
125.     write_data_to_binary();
126.     read_data_from_binary_file();
127.     find();
128.     cout << "_____\n";
129.
130.     change();
131.     print_all_data();
132.     cout << "_____\n";
133.
134.     del();
135.     print_all_data();
136. }
```

Результат выполнения программы:

```
Number : 1234567
Name : Ivan
Surname : Ivanov
Second Name : Ivanov
Address : Perm

_____
Number : 7654321
Name : Alex
Surname : Alexandrov
Second Name : Sergeevich
Address : Moscow

_____
_____
Input number of account, which you want to find: 1234567

Number : 1234567
Name : Ivan
Surname : Ivanov
Second Name : Ivanov
Address : Perm

_____
_____
Input number of account, which you want to change : 1234567

Input new name : Petr

Number : 1234567
Name : Petr
Surname : Ivanov
Second Name : Ivanov
Address : Perm

_____
Number : 7654321
Name : Alex
Surname : Alexandrov
Second Name : Sergeevich
Address : Moscow

_____
_____
Input number of account, which you want to delete : 7654321

Number : 1234567
Name : Petr
Surname : Ivanov
Second Name : Ivanov
Address : Perm
```

Практическая работа №13

Вариант №1 – Основные алгоритмы работы с графами.

Автор: Николаев-Аксенов И. С.

Группа: ИКБО-20-19

Задание:

Составить программу реализации алгоритма Крускала построения остовного дерева минимального веса.

Выбрать и реализовать способ представления графа в памяти.

Предусмотреть ввод с клавиатуры произвольного графа.

Разработать доступный способ (форму) вывода результирующего дерева на экран монитора.

Провести тестовый прогон программы для заданного графа в соответствии с индивидуальным заданием

Код программы:

Graph.java:

```
1.  public class Graph implements Comparable<Graph> {
2.      public int A;
3.      public int mass;
4.      public int B;
5.
6.      Graph(int A, int mass, int B) {
7.          this.A=A;
8.          this.B=B;
9.          this.mass=mass;
10.     }
11.
12.     @Override
13.     public int compareTo(Graph o) {
14.         if(mass!=o.mass){
15.             return mass<o.mass? -1:1;
16.         }
17.         return 0;
18.     }
19. }
```

SetGraph.java:

```
1.  public class SetGraph {
2.      int[] number, rang;
3.
4.      SetGraph(int size) {
5.          number=new int[size];
6.          rang=new int[size];
7.          for(int i=0;i<size;++i){
8.              number[i]=i;
9.          }
10.     }
11.
```

```
    int set(int x) {
        return x==number[x]? x:(number[x]=set(number[x]));
    }

    boolean merge(int A, int B){
        if(set(A)==set(B))
            return false;

        if(rang[A]<rang[B])
            number[A]=B;

        else {
            number[B]=A;
            if(rang[A]==rang[B])
                rang[A]++;
        }
        return true;
    }
}
```

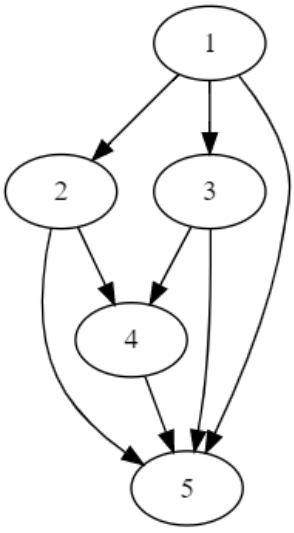
StartKruskalsAlgorithm.java:

```
import java.util.*;

public class StartKruskalsAlgorithm {
    public static int KruskalAlgorithm(ArrayList<Graph> graph) {
        SetGraph union = new SetGraph(graph.size()+1);
        Collections.sort(graph);

        ArrayList<Graph> buff = new ArrayList<>();
        for(Graph i: graph) {
            if(union.merge(i.A,i.B)){
                buff.add(i);
            }
        }
        graph.clear();
        graph.addAll(buff);
        return 0;
    }

    public static void main(String[] args) {
        ArrayList<Graph> graph = new ArrayList<>();
        Scanner scanner = new Scanner(System.in);
        int a,b,m;

        System.out.println("Ввод ребер графа: ");
        while(true){
            a = scanner.nextInt();
            if(a==0) break;
            m = scanner.nextInt();
            b = scanner.nextInt();
            graph.add(new Graph(a,m,b));
        }
        System.out.println("Исходный граф: ");
        for (Graph item : graph) {
            System.out.println(item.A + "->" + item.B);
        }
        System.out.println(KruskalAlgorithm(graph));
        System.out.println("Результат: ");
        for (Graph value : graph) {
            System.out.println(value.A + "->" + value.B);
        }
        System.out.println("В данной работе используется язык graphviz. Посмотреть граф
можно на сайте graphvix.org.");
    }
}
```

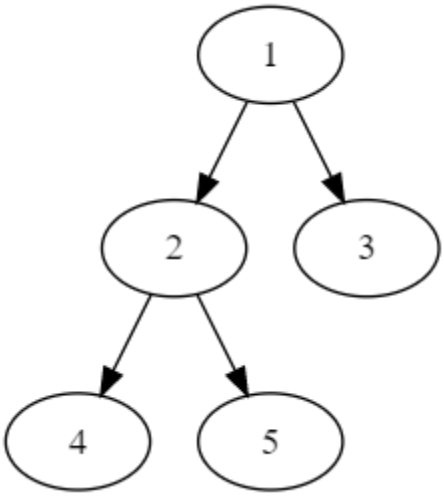
Результат выполнения программы:

```
Ввод ребер графа:
1 1 2
1 10 5
1 2 3
2 6 5
3 7 5
2 3 4
4 11 5
3 4 4
0
Исходный граф:
1->2
1->5
1->3
2->5
3->5
2->4
4->5
3->4
0
Результат:
1->2
1->3
2->4
2->5
```

Исходный граф:



Результат:



Практическая работа №14

Алгоритмы сжатия и кодирования данных.

Автор: Николаев-Аксенов И. С.

Группа: ИКБО-20-19

Код программы:

```
#include <iostream>
#include <vector>
#include <list>
#include <string>
#include <algorithm>
#include <clocale>
#include <map>
#include <iomanip>
#include <queue>
using namespace std;
map<char, string> codes;
map<char, int> freq;

struct MinHeapNode
{
    char data;
    int freq;
    MinHeapNode* left, * right;

    MinHeapNode(char data, int freq)
    {
        left = right = NULL;
        this->data = data;
        this->freq = freq;
    }
};


struct compare
{
    bool operator()(MinHeapNode* l, MinHeapNode* r)
    {
        return (l->freq > r->freq);
    }

};



void storeCodes(struct MinHeapNode* root, string str)
{
    if (root == NULL)
        return;
    if (root->data != '$')
        codes[root->data] = str;
    storeCodes(root->left, str + "0");
    storeCodes(root->right, str + "1");
}


priority_queue<MinHeapNode*, vector<MinHeapNode*>, compare> minHeap;
template<typename T> void print_queue(T& q,int size)
{
    priority_queue<MinHeapNode*, vector<MinHeapNode*>, compare> q1=q;
    priority_queue<MinHeapNode*, vector<MinHeapNode*>, compare> q2 = q;

    cout << "Алфавит: ";
```

```
58.     while(!q.empty())
59.     {
60.         if (q.top() != NULL)
61.             cout << setw(4) << q.top()->data << " ";
62.
63.         q.pop();
64.     }
65.
66.     cout << endl << "Количество вхождений: ";
67.     while (!q1.empty())
68.     {
69.         if (q1.top() != NULL)
70.             cout << setw(4) << q1.top()->freq << " ";
71.
72.         q1.pop();
73.     }
74.
75.     cout << endl << "Вероятность: ";
76.     while (!q2.empty())
77.     {
78.         if (q2.top() != NULL)
79.             cout << setw(4) << float(q2.top()->freq)/size << " ";
80.
81.         q2.pop();
82.     }
83.
84.     cout << '\n';
85. }
86. void HuffmanCodes(int size)
87. {
88.     struct MinHeapNode* left, * right, * top;
89.
90.     for (map<char, int>::iterator v = freq.begin(); v != freq.end(); v++)
91.         minHeap.push(new MinHeapNode(v->first, v->second));
92.     priority_queue<MinHeapNode*, vector<MinHeapNode*>, compare> minHeap2=minHeap;
93.     print_queue(minHeap2,size);
94.
95.     while (minHeap.size() != 1)
96.     {
97.         left = minHeap.top();
98.         minHeap.pop();
99.         right = minHeap.top();
100.          minHeap.pop();
101.          top = new MinHeapNode('$', left->freq + right->freq);
102.          top->left = left;
103.          top->right = right;
104.          minHeap.push(top);
105.      }
106.      storeCodes(minHeap.top(), "");
107. }
108.
109. void calcFreq(string str, int n)
110. {
111.     for (int i = 0; i < str.size(); i++)
112.         freq[str[i]]++;
113. }
114.
115. string haffman_code(string input)
116. {
117.     string encodedString;
118.     for (auto i : input)
119.         encodedString += codes[i];
120.     return encodedString;
121. }
122.
123. void print_table(string str)
124. {
125.     cout << "Алфавит: ";
126.     for (auto item : freq)
```

```
127.         cout << setw(4) << item.first << " ";
128.
129.     cout << endl << "Количество вхождений: ";
130.     for (auto item : freq)
131.         cout << setw(4) << item.second << " ";
132.
133.     cout << endl << "Вероятность: ";
134.     for (auto item : freq) {
135.         cout.setf(std::ios::fixed);
136.         cout << setprecision(2) << float(item.second)/str.length() << " ";
137.     }
138.
139.     cout << endl;
140.     cout << endl;
141. }
142.
143. void sviaz_codov()
144. {
145.     for (auto s : codes)
146.         cout << s.first << ": " << s.second << endl;
147. }
148.
149. int dec2bin(int num)
150. {
151.     int bin = 0, k = 1;
152.     while (num)
153.     {
154.         bin += (num % 2) * k;
155.         k *= 10;
156.         num /= 2;
157.     }
158.     return bin;
159. }
160.
161. string ascii_code(string input) {
162.     string asci = "";
163.     for (int i = 0; i < input.size(); ++i) {
164.         asci += to_string(dec2bin((int(input[i]))));
165.     }
166.     return asci;
167. }
168.
169. void results(string a) {
170.     cout << "Коды символов: " << endl; sviaz_codov();
171.     cout << "Код по Хаффману: " << haffman_code(a) << endl;
172.     cout << "Длина кода по алгоритму Хаффмана: " << haffman_code(a).length() << endl;
173.     cout << "Код по ASCII: " << ascii_code(a) << endl;
174.     cout << "Длина кода по ASCII: " << ascii_code(a).length() << endl;
175.     cout << "Дисперсия " << ((float)haffman_code(a).length() / ascii_code(a).length())
    << endl;
176. }
177.
178. int main()
179. {
180.     setlocale(LC_ALL, "Russian");
181.     string str = "nikolaevaxenov ivan sergeevich";
182.     calcFreq(str, str.length());
183.     print_table(str);
184.     HuffmanCodes(str.length());
185.     results(str);
186.     return 0;
187. }
```

Результат выполнения программы:

```
Алфавит:          a    c    e    g    h    i    k    l    n    o    r    s    v    x
Количество вхождений:    2    3    1    5    1    1    3    1    1    3    2    1    1    4    1
Вероятность: 0.07 0.10 0.03 0.17 0.03 0.03 0.10 0.03 0.03 0.10 0.07 0.03 0.03 0.13 0.03

Алфавит:    c    h    x    r    s    g    k    l         o    i    n    a    v    e
Количество вхождений:    1    1    1    1    1    1    1    1    2    2    3    3    3    4    5
Вероятность: 0.03 0.03 0.03 0.03 0.03 0.03 0.03 0.03 0.07 0.07 0.10 0.10 0.10 0.13 0.17
Коды символов:
 : 1001
a: 000
c: 10000
e: 111
g: 10101
h: 10001
i: 010
k: 11000
l: 11001
n: 001
o: 1101
r: 10111
s: 10100
v: 011
x: 10110
Код по Хаффману: 00101011000110111001000111011000101101110011101011100101001100000110011010011110111101011111
10110101000010001
Длина кода по алгоритму Хаффмана: 110
Код по ASCII: 11011101101001110101111011111101100110000111001011110110110000111110001100101110111011011111110
10100000110100111101101100001110111010000011100111100101111001011100101111011011010011100011110100
00
Длина кода по ASCII: 208
Дисперсия 0.53
```

Практическая работа №15

Вариант №2 – Расстановка скобок.

Автор: Николаев-Аксенов И. С.

Группа: ИКБО-20-19

Код программы:

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

void printParenthesis(int i, int j, int n, int *bracket, char &name)
{
    if (i == j)
    {
        cout << name++;
        return;
    }

    cout << "(";

    printParenthesis(i, *((bracket + j * n) + i), n, bracket, name);

    printParenthesis(*((bracket + j * n) + i) + 1, j, n, bracket, name);
    cout << ")";
}

void matrixChainOrder(int p[], int n)
{
    int min[n][n];
    int max[n][n];

    for (int i = 1; i < n; i++)
    {
        min[i][i] = 0;
        max[i][i] = 0;
    }

    for (int L = 2; L < n; L++)
    {
        for (int i = 1; i < n - L + 1; i++)
        {
            int j = i + L - 1;
            min[i][j] = INT_MAX;
            max[i][j] = INT_MIN;
            for (int k = i; k <= j - 1; k++)
            {
                int q = min[i][k] + min[k + 1][j] + p[i - 1] * p[k] * p[j];
                if (q < min[i][j])
                {
                    min[i][j] = q;
                    min[j][i] = k;
                }

                if (q >= max[i][j])
                {
                    max[i][j] = q;
                    max[j][i] = k;
                }
            }
        }
    }

    char matrixName = 'A';
```

```
58.     cout << "Оптимальная расстановка скобок: ";
59.     printParenthesis(1, n - 1, n, (int *) min, matrixName);
60.     cout << "\nМинимальное количество скалярных операций : " << min[1][n - 1] << endl;
61.
62.     matrixName = 'A';
63.     cout << "\nНеоптимальная расстановка скобок: ";
64.     printParenthesis(1, n - 1, n, (int *) max, matrixName);
65.     cout << "\nМаксимальное количество скалярных операций : " << max[1][n - 1];
66. }
67.
68.
69. int main()
70. {
71.     setlocale(LC_ALL, "Russian");
72.
73.     int arr[] = {5, 10, 3, 12, 5, 50, 6};
74.
75.     int n = sizeof(arr) / sizeof(arr[0]);
76.     matrixChainOrder(arr, n);
77.
78.     return 0;
79. }
```

Результат выполнения программы:

```
Оптимальная расстановка скобок: ((AB)((CD)(EF)))
Минимальное количество скалярных операций : 2010

Неоптимальная расстановка скобок: (((A(BC))(DE))F)
Максимальное количество скалярных операций : 3155
```

Практическая работа №16

Вариант №6 – Реализовать задачу о рюкзаке методом ветвей и границ.

Автор: Николаев-Аксенов И. С.

Группа: ИКБО-20-19

Код программы:

```cpp
#include <iostream>
#include <algorithm>
#include <queue>
using namespace std;

struct Item
{
    float weight;
    int value;
};

struct Node
{
    int level, profit, bound;
    float weight;
};

bool cmp(Item a, Item b)
{
    double r1 = (double) a.value / a.weight;
    double r2 = (double) b.value / b.weight;
    return r1 > r2;
}

int bound(Node u, int n, int knapsackWeight, Item arr[])
{
    if (u.weight >= knapsackWeight)
        return 0;

    int profit_bound = u.profit;

    int j = u.level + 1;
    int totalWeight = u.weight;

    while ((j < n) && (totalWeight + arr[j].weight <= knapsackWeight))
    {
        totalWeight += arr[j].weight;
        profit_bound += arr[j].value;
        j++;
    }

    if (j < n)
        profit_bound += (knapsackWeight - totalWeight) * arr[j].value / arr[j].weight;

    return profit_bound;
}

int knapsack(int W, Item arr[], int n)
{
    sort(arr, arr + n, cmp);

    queue<Node> Q;
    Node u, v;

    u.level = -1;
    u.profit = u.weight = 0;
    Q.push(u);
```

```
58.
59.     int maxProfit = 0;
60.     while (!Q.empty())
61.     {
62.         u = Q.front();
63.         Q.pop();
64.
65.         if (u.level == -1)
66.             v.level = 0;
67.
68.         if (u.level == n - 1)
69.             continue;
70.
71.         v.level = u.level + 1;
72.         v.weight = u.weight + arr[v.level].weight;
73.         v.profit = u.profit + arr[v.level].value;
74.
75.         if (v.weight <= W && v.profit > maxProfit)
76.             maxProfit = v.profit;
77.
78.         v.bound = bound(v, n, W, arr);
79.
80.         if (v.bound > maxProfit)
81.             Q.push(v);
82.
83.         v.weight = u.weight;
84.         v.profit = u.profit;
85.         v.bound = bound(v, n, W, arr);
86.         if (v.bound > maxProfit)
87.             Q.push(v);
88.     }
89.
90.     return maxProfit;
91. }
92.
93. int main() {
94.     setlocale(LC_ALL, "Russian");
95.
96.     int W = 10;
97.     Item arr[] = {{2,    40}, {3.14, 50}, {1.98, 120}, {5,    95}, {3,    30}};
98.     int n = sizeof(arr) / sizeof(arr[0]);
99.
100.     cout << "Маскимальная стоимость: " << knapsack(W, arr, n);
101.
102.     return 0;
103. }
```

Результат выполнения программы: