**Лабораторная работа по курсу «Инструментальные средства и технологии программирования» по теме: Лямбда-функции**

Реализовать посредством лямбда-выражений «калькулятор», считающий количество вариантов получения одного числа из другого с помощью набора операций.

Параметры задания:

1. Два числа: необходимо из одного числа получить второе с помощью нескольких арифметических операций.
2. Набор арифметических операций.
3. Ограничения (т.е. через какие числа должна проходить «траектория» получения нового числа.

Порядок выполнения задания:

1. Создать «прямую» программу получения из одного числа другого. Программа должна выводить все «траектории» получения из одного числа другого, а также считать количество вариантов, которые просчитывала программа.
2. Создать программу с ограничениями («траектория» получения результата должна проходить через числа-ограничения (третий параметр задания).)
3. Создать «обратную» программу: заменить все команды на обратные (например, умножение на 2, заменяется делением на 2). Рассчитать количество успешных вариантов, сравнить с предыдущим вариантом.
4. Создать программу, использующую элементы исследования операций (вычисленные количества операций запоминаются в вектор или словарь (map), т.е. запоминается фазовое пространство. Вывести вектор фазового пространства.

Пример программы, использующей лямбда-функции для вычисления рекурсивного выражения:

```
#include <iostream>
#include <functional>
#include <string>
#include <map>

using namespace std;

const int start=2, fin=8; // Начальное и конечное числа

int main()
{
  // Прямой ход расчета
  int breanchCount=0; // Счётчик просмотренных веток дерева
  function<int (int, int,string)> calc = [&calc, &breanhcCount](int x, int fin, string way)
   {
      breanchCount++;
      if(x<fin) return calc(x+1,fin,way+" +1") + calc(x*2,fin,way +" *2");
      else
         if(x==fin) {
            //cout<<way<<endl;
            return 1;
         }
      return 0;
   };
   cout<<"Forward\nCount: "<<calc(start, fin, "")<<endl;
   cout<<"breanchCount:"<<breanchCount<<endl;
   breanchCount=0;
// Расчет обратным ходом
   function<int (int, int,string)> calcBack = [&calcBack, &breanhcCount](int x, int fin, string way)
```

```
    {
        breanchCount++;
        if(x>fin)
            {
                int t=calcBack(x-1,fin,"+1"+way);
                if (x%2==0) t+=calcBack(x/2,fin,+"*2"+way);
                return t;
            }
        else
            if(x==fin) {
                return 1;
                //cout<<way<<endl;
            }
    }
    cout<<"Back\nCount: "<<calcBack(fin, start, "")<<endl;
    cout<<"breanchCount:"<<breanhcCount<<endl;

    // Расчет методом исследования операций
    int breanchCount=0;
    map<int,int> Space;

    function<int (int, int)> calcOp = [&Space, &calcOp, &breanhcCount](int x, int fin)
    {
        breanchCount++;
        if(Space.count(x)) return Space[x];
        if(x==fin)
        {
            Space[x]=1;
            return 1;
        }
        if(x<fin)
            {
            Space[x]= calcOp(x+1,fin)+calcOp(x*2,fin);
            return Space[x];
            }
        return 0;
    };
    cout<<"OR\nCount: "<<calcOp(start, fin)<<endl;
    cout<<"BreanchCount: "<<breanchCount<<endl;

// Вывод вектора фазового пространства
//    for(auto i: Space)
//        cout<<i.first<<" "<<i.second<<endl;

    return 0;
}
```

Варианты заданий

| | | | |
|---|---|---|---|
| 1. | 3 28 | +1 *3 | 14 или 20 |
| 2. | 2 29 | +3 *2 | 12 или 28 |
| 3. | 5 32 | +4 *2 | 20 или 26 |
| 4. | 4 25 | +1 *2 | 10 или 20 |
| 5. | 2 30 | +1 ^2 | 8 или 9 |
| 6. | 3 24 | +2 *2 | 6 или 14 |
| 7. | 2 26 | +2 +5 | 6 и 12 |
| 8. | 4 27 | +1 + (*2-1) +1 | 7 или 17 |
| 9. | 3 25 | +1 (*2+1) | 8 или 16 |

10. 4 24 +2 (*2-2) 9 или 18
11. 2 27 +1 (*2+1) 8 или 15
12. 4 28 +4 (*2-1) 7 или 17
13. 3 28 +1 (*2+2) 8 или 16
14. 2 23 +2 (*2-1) 9 или 18
15. 2 27 +3 (*2+1) 8 или 14
16. 4 25 +2 *3 12 или 20
17. 2 30 +2 ^2 8 или 12
18. 3 23 +2 (*3-1) 9 или 13
19. 2 26 +1 +5 6 и 12
20. 3 28 +2 *3 13 или 21
21. 2 29 +3 (*2+1) 14 или 28
22. 5 30 +3 *2 18 или 25
23. 3 28 +2 (*3+1) 15 или 21
24. 2 28 +4 *2 12 или 22
25. 5 40 +3 *4 20 или 25
26. 2 60 +1 ^2 8 или 33
27. 4 29 +2 (*2-1) 9 или 19
28. 2 27 +2 (*2+1) 9 или 17
29. 3 32 +1 *3 13 или 21
30. 3 28 +2 *2 15 или 20