Разбор задач олимпиады «Третья командная олимпиада» 01.11.2014г.

ПРОФИТ
тренинг-центр

Задача A. Станки

Минимальное время будет получено в том случае, если одновременно использовать оба станка, и будет равно количеству деталей, поделённое на суммарную производительность станков.

Но надо ещё округлить до целого в большую сторону. Мы неоднократно писали, как это делать. Например в задаче «Ремонт»:
https://vk.com/doc-57951380_227853695

В таком решении подводным камнем может быть переполнение типа *int*. Поэтому следует использовать приведение к типу *long long*.

Решение на языке C++

```cpp
#include <iostream>
#include <cstdio>

using namespace std;

int main() {
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);

    int a, b, c;
    cin >> a >> b >> c;
    cout << (a-1LL+b+c)/(b+c);
    return 0;
}
```

Задача B. Распродажа

Это обычная задача на умение работы с процентами. Но основная сложность в области программирования свелась к округлению результата до целого числа.

Вещественное число *A* можно округлить до целого (по правилам математики), используя следующий трюк:

int N = (int)(A + 0.5 + EPS)

Где *EPS* – это некоторое небольшое число. Методом проб в этом задаче в качестве *EPS* можно было взять 0.001.

Решение на языке C++

```cpp
#include <iostream>
#include <cstdio>

#define EPS 1e-3

using namespace std;

int main() {
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);

    int n, a, b;
    cin >> n >> a >> b;
    n = (int)(n*(1+a/100.)+0.5+EPS);
    n = (int)(n*(1-b/100.)+0.5+EPS);
    cout << n;
    return 0;
}
```

Задача C. Последовательность

Для решения задачи будем последовательно считывать данные и сразу же обрабатывать. Если число делится на *m* без остатка, тогда надо попробовать обновить значения максимума и минимума.

Начинающие программисты могут столкнуться с проблемой выбора начальных значений для максимума и минимума. Рассмотрим решение для максимума (для минимума всё аналогично). Обычно начальным значением выбирается очень маленькое число. В этой же задаче был занят весь диапазон типа *int*, поэтому следовало выбрать самое маленькое число, которое помещается в этот тип данных. Либо воспользоваться типом *long long* и выбрать число на несколько порядков меньше.

Решение на языке C++

```cpp
#include <iostream>
#include <cstdio>

#define LLINF (9223372036854775807LL)

using namespace std;

int main() {
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);

    int n, m, a;
    long long mx = -LLINF, mn = LLINF;
    cin >> n >> m;
    for(int i=0; i<n; ++i){
        cin >> a;
        if(a%m == 0){
            mx = (mx > a ? mx : a);
            mn = (mn < a ? mn : a);
        }
    }
    if(mn == LLINF) cout << "NO";
    else cout << mn << " " << mx;
    return 0;
}
```

Задача D. Обратное число

Заметим, что получить обратное число можно, применив операцию «Исключающее ИЛИ» (XOR) с числом из всех единичек (в двоичном представлении).

Например дано число 13. В двоичной записи это будет 1101_2. Если сделать XOR с числом 1111_2, то получится 0010_2.

Осталось узнать количество разрядов в двоичном представлении числа. Это можно сделать с помощью битового сдвига единички влево.

Более подробно про битовые операции можно прочитать в сборнике «Справочник спортивного программиста (часть 1)»: https://vk.com/topic-57951380_30084443

Решение на языке C++

```cpp
#include <iostream>
#include <cstdio>

using namespace std;

int main() {
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);

    long long d, p = 1;
    cin >> d;
    while(p<=d) p <<= 1;
    cout << (d^p-1);
    return 0;
}
```

Задача E. Шифровка

Так как алгоритм шифрования слова повторяется и для шифрования всей фразы, то рассмотрим лишь дешифровку фразы.

Будем идти по словам двумя указателями (один с начала, другой с конца) и добавлять слова, на которые они указывают, в результат. Но нужно учесть два случая.

Если количество слов чётное, то в начало строки сначала добавляется левое слово, а затем правое. Если же количество слов нечётное, то наоборот. Либо, как в решении ниже, можно отдельно добавить последнее слово, приведя случай к чётному варианту.

Решение на языке C++

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

string f(string s){
    string res = "", t1, t2;
    int i=0, j=s.size()-1;
    if(s.size() & 1) res += s[j--];
    while(i<j) res = (t1 = s[j--]) + (t2 = s[i++]) + res;
    return res;
}

int main() {
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);

    vector<string> v;
    string s;
    while(cin >> s) v.push_back(f(s));

    s = "";
    int i=0, j=v.size()-1;
    if(v.size() & 1) s += v[j--];
    while(i<j) s = v[j--] +" "+ v[i++] +" "+ s;
    cout << s;
    return 0;
}
```

Задача F. Формула

Авторское решение этой задачи короткое и элегантное, но трудно экстраполируется на другие подобные задачи. Мы же приводим решение с помощью стандартного разбора выражений, который выполняет рекурсивная функция. В зависимости от постановки задачи, в эту функцию можно добавить обработку любых (арифметических, битовых, логических и других) операций.

Функция принимает отрезок строки, для которого надо вычислить значения, и возвращает результат вычисления. А сама в обратном порядке приоритетов операций разбивает выражение на две части и вызывается рекурсивно.

Решение на языке C++

```cpp
#include <iostream>
#include <cstdio>
#include <string>

using namespace std;

string s;
int rec(int l, int r){

    if(l==r) return s[l]-'0';

    int m = l+2, c=0;
    for(; m<=r; ++m)
        if(s[m]==',' && !c) break;
        else if(s[m]=='(') ++c;
        else if(s[m]==')') --c;

    if(s[l]=='M') return max(rec(l+2, m-1), rec(m+1, r-1));
    return min(rec(l+2, m-1), rec(m+1, r-1));
}

int main() {
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);

    cin >> s;
    cout << rec(0, s.size()-1);
    return 0;
}
```

Задача G. Экзамены

Задача легко решается с использованием STL. Заведём вектор пар, в котором будем хранить сумму баллов и фамилию каждого человека. Тогда функцией sort из библиотеки algorithm можно его отсортировать. Для сортировки в обратном порядке можно использовать реверсивные итераторы.

Осталось лишь вывести информацию о нужном человеке.

Решение на языке C++

```cpp
#include <iostream>
#include <cstdio>
#include <string>
#include <vector>
#include <algorithm>

using namespace std;

int main() {
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);

    int n, m, a, b, c;
    cin >> n >> m;
    vector<pair<int, string> > v;
    for(int i=0; i<n; ++i){
        string s;
        cin >> s >> a >> b >> c;
        v.push_back(make_pair(a+b+c, s));
    }
    sort(v.rbegin(), v.rend());
    cout << v[m-1].second << " " << v[m-1].first;
    return 0;
}
```

Задача H. Отрезок

Координаты точки C (середины отрезка) вычисляются по формулам

$$\left(\frac{A_x+B_x}{2};\frac{A_y+B_y}{2}\right).$$

Отсюда можно выразить координаты точки B: $\left(2\cdot C_x-A_x;2\cdot C_y—A_y\right)$.

Решение на языке C++

```cpp
#include <iostream>
#include <cstdio>

using namespace std;

int main() {
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);

    int x1, x2, y1, y2;
    cin >> x1 >> y1 >> x2 >> y2;
    cout << 2*x2-x1 << " " << 2*y2-y1;
    return 0;
}
```

Задача I. Лабиринт

Для решения задачи воспользуемся алгоритмом поиска в ширину. Для этого служит функция *bfs*, которая в каждой клетке массива отмечает, за сколько ходов можно до неё добраться.

Тогда сначала надо запустить эту функцию из исходной точки героя. После проверки, что выход не найден (а если найден, то уже можно вывести ответ), сразу же можно узнать есть ли путь до ключа.

Если путь до ключа есть, то в исходном массиве надо все двери заменить на пустые клетки и запустить *bfs* от ключа.

Решение на языке C++

```cpp
#include <iostream>
#include <cstdio>
#include <memory.h>
#include <queue>

#define INF (2147483647)
using namespace std;

int n, m;
char s[1009][1009];
int mm[1009][1009];

void bfs(int x, int y){
    memset(mm, 0, sizeof(mm));
    queue<pair<int,int> > q;
    q.push(make_pair(x, y));
    int d[4][2] = {{0,1},{0,-1},{1,0},{-1,0}};
    mm[x][y] = 1;
    while(!q.empty()){
        pair<int,int> p = q.front();
        q.pop();
        x = p.first; y = p.second;
        for(int i=0; i<4; ++i)
            if(x+d[i][0]>=0 && x+d[i][0]<n &&
               y+d[i][1]>=0 && y+d[i][1]<m &&
               s[x+d[i][0]][y+d[i][1]]=='.' && !mm[x+d[i][0]][y+d[i][1]])
                mm[x+d[i][0]][y+d[i][1]] = mm[x][y] + 1,
                q.push(make_pair(x+d[i][0], y+d[i][1]));
    }
}
```

```
30  int main() {
31      freopen("input.txt", "r", stdin);
32      freopen("output.txt", "w", stdout);
33
34      cin >> n >> m;
35      int ik, jk, it, jt;
36      for(int i=0; i<n; ++i){
37          cin >> s[i];
38          for(int j=0; j<m; ++j){
39              if(s[i][j] == 'K') ik = i, jk = j, s[i][j] = '.';
40              if(s[i][j] == 'T') it = i, jt = j, s[i][j] = '.';
41          }
42      }
43      bfs(it, jt);
44
45      int mn = INF;
46      for(int i=0; i<n; ++i)
47          for(int j=0; j<m; ++j)
48              if((!i || !j || i==n-1 || j==m-1) && mm[i][j])
49                  mn = min(mn, mm[i][j]);
50
51      if(mn < INF){
52          cout << mn;
53          return 0;
54      }else cout << "No way ";
55
56      if(!mm[ik][jk]){
57          cout << "No key";
58          return 0;
59      }else cout << mm[ik][jk]-1 << " ";
```

```
61          for(int i=0; i<n; ++i)
62              for(int j=0; j<m; ++j)
63                  if(s[i][j] == '-') s[i][j] = '.';
64
65          bfs(ik, jk);
66
67          mn = INF;
68          for(int i=0; i<n; ++i)
69              for(int j=0; j<m; ++j)
70                  if((!i || !j || i==n-1 || j==m-1) && mm[i][j])
71                      mn = min(mn, mm[i][j]);
72
73          if(mn < INF) cout << mn;
74          else cout << "No way";
75
76          return 0;
77      }
```

Задача J. Квадрат

Рассмотрим одномерный случай. Если взять отрезок целой длины *L*, то он может покрыть *L+1* точку с целыми координатами. Для этого оба его конца должны быть в целых координатах (это возможно, так как длина отрезка – целое число).

Тогда в двумерном варианте, мы по каждой из координат можем закрыть по *L+1* точке. И общее количество закрытых точек будет равно $(L+1)^2$.

Решение на языке C++

s

```cpp
#include <iostream>
#include <cstdio>

using namespace std;

int main() {
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);

    int a;
    cin >> a;
    cout << (a+1)*(a+1);
    return 0;
}
```

Тренинг-центр «Профит»

Объявляется набор на курсы по информатике и программированию:

— "Программирование для начинающих",
— "Программирование на языке C++",
— "Олимпиадное программирование",
— "ЕГЭ по информатике".

Занятия ведут лучшие программисты Восточной Сибири, подготовившие призёров и победителей регионального этапа Всероссийской олимпиады школьников по информатике. А сам тренинг-центр является региональным сектором Открытого кубка по программированию.

Наша группа во ВКонтакте: https://vk.com/profit_krsk

ТРЕНИНГ-ЦЕНТР
ПРОФИТ