**Федеральное Агентство по Образованию**

Государственное образовательное учреждение высшего профессионального образования

**Санкт-Петербургский Государственный Электротехнический Университет «ЛЭТИ» им. В.И.Ульянова (Ленина)**

**(СПБГЭТУ)**

Кафедра МОЭВМ

Отчет по лабораторной работе №2

по дисциплине «Анализ и проектирование алгоритмов»

Выполнил: Черницын Н. А.

Факультет: КТИ

Группа: 9301

Проверил: Казаков Б.Б.

#### Санкт-Петербург,

# 2013

# ***Формулировка задания:*** Кубик Сома. Следующие семь элементов можно сложить вместе многими способами для того, чтобы составить куб размером 3х3х3.

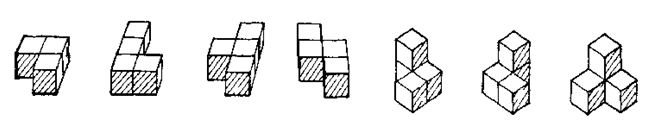


Рисунок 1 – фигуры кубика Сома

Найдите число неэквивалентных (относительно вращения и/или) отражения решений.

***Теоретическая часть:***

Кубики сома — это головоломка, состоящая из 7 элементов разной формы. Из этих элементов собирают куб размером 3x3x3, всего существует 240 разных способов сборки.

Кубики сома Пит Хейн придумал во время лекции Вернера Гейзенберга по квантовой механике. Пока знаменитый физик говорил о пространстве, разрезанном на кубики, живое воображение Пита Хейна подсказало ему формулировку любопытной геометрической теоремы: если взять все неправильные фигуры, которые составлены из трех или четырех кубиков, склеенных между собой гранями, то из них можно составить один кубик большего размера.

Простейшая неправильна фигура - «неправильная» в том смысле, что на ней имеются выступы и впадины, - получаются, если склеить три или четыре кубика так, как показано на рисунке 1.

Хейн обратил внимание на то, что, склеивая два куба, мы увеличиваем протяженность тела лишь в одном направлении. Чтобы увеличить протяженность тела в другом направлении, нам нужен еще один, третий кубик. Четыре кубика позволят увеличить протяженность тела в трех направлениях. Поскольку, даже взяв пять кубиков, мы не увеличим размерность фигуры до четырех, набор кубиков сома разумно ограничить семью фигурами. Совершенно неожиданно выяснилось, что из этих семи элементов можно сложить один большой куб.

***Алгоритм работы программы:***

Число клеток в коробке совершенно не больше, чем число бит в одном из представлений целых чисел, именно это представление было выбрано в качестве основного: как текущее состояние поля, так и каждое положение каждой фигуры хранится в виде битовой маски int.

Количество ориентаций каждой фигуры — не более 12, количество положений при данной ориентации — не более 18. Итого, возможно не более 216 положений каждой фигуры , поэтому мы будем хранить их в памяти.

Выберем в группе поворотов куба две и будем применять их к одному состоянию, пока результаты не стабилизируются.

Преобразования: (x,y,z) -> (y,z,x) и (x,y,z) -> (-x,z,y).

Функция Turn() (смотри листинг) выполняет один из двух поворотов. Константы XHIGH, XLOW и т.п. соответствуют граням куба, и позволяют определить, можно ли двигать фигуру в ту или иную сторону.

На каждом шаге мы можем либо выбрать по порядку клетку и перебрать все свободные фигуры, отсортированные по минимальной клетке, которыми ее можно закрыть.

Положения фигур сохраняются в массиве FState, и функция Print() выводит на экран найденное решение.

***Листинг:***

* *Main.cpp*

#include <stdlib.h>

#include <time.h>

#include <iostream>

const int XLOW=0x1249249;

const int XHIGH=0x4924924;

const int YHIGH=0x70381C0;

const int ZHIGH=0x7FC0000;

struct Fig{

int NF; //размер Arr

int \*Arr;

int Ind[28];

Fig() : Arr(NULL) {}

void Init(int fig0);

void Init(int nf,int \*arr);

~Fig(){ delete Arr; }

};

int arr0[648];

int LL[4]=

{

0x7800000,

0x0F00000,

0x003C000,

0x001E000

};

const int NFig=7;

int Figs0[NFig]=

{

0xF,

0x17,

0x33,

0xB,

0x20B,

0x213,

0x219

};

Fig Figs[NFig];

int FState[NFig];

int NSolve=0; //количество решений

int NBack=0; //количество возвратов

long TStart; //подсчет времени работы алгоритма

int Turn(int v, int rt);

void Print();

void Solve(int x,int mb,int z);

int Turn(int v,int rt){

int res=0;

for(int a=0;a<27;a++){

int b=rt==0 ? (a/3)+(a%3)\*9 : (2-a%3)+(a/9)\*3+((a/3)%3)\*9;

if((v>>a)&1)

res|=1<<b;

}

if(rt){

while((res&XLOW)==0) res>>=1;

}

return res;

}

extern "C" int i64cmp(const void \*a,const void \*b)

{

int A=\*(const int\*)a,B=\*(const int\*)b;

return A<B ? -1 : A==B ? 0 : 1;

}

void Fig::Init(int fig0)

{

int p=0,q=1;

arr0[0]=fig0;

while(p<q)

{

int a=arr0[p++];

for(int u=0;u<2;u++)

{

int b=Turn(a,u);

int i;

for(i=0;i<q;i++)

if(arr0[i]==b)

break;

if(i==q)

arr0[q++]=b;

}

}

for(p=0;p<q;p++)

if((arr0[p]&XHIGH)==0)

arr0[q++]=arr0[p]<<1;

for(p=0;p<q;p++)

if((arr0[p]&YHIGH)==0)

arr0[q++]=arr0[p]<<3;

for(p=0;p<q;p++)

if((arr0[p]&ZHIGH)==0)

arr0[q++]=arr0[p]<<9;

Init(q,arr0);

}

void Fig::Init(int q,int \*arr)

{

NF=q;

Arr=new int[q];

for(int p=0;p<q;p++)

Arr[p]=arr[p]; //копирование arr в Arr

qsort(Arr,q,sizeof(int),i64cmp);

int j=0;

for(int p=0; p<q; p++)

while(Arr[p]&(-1<<j))

Ind[j++]=p;

while(j<=27)

Ind[j++]=q;

for(int i=0; i<27; i++)

std::cout<<Ind[i]<<" ";

std::cout<<std::endl;

}

void Print()

{

std::cout<<"Решение №"<<NSolve<<std::endl;

for(int i=0;i<27;i++) //перебор всех битов куба

{

int l=1<<i;

int f;

for(f=0;f<NFig;f++)

{

if(Figs[f].Arr[FState[f]]&l)

{

std::cout<<(char)('A'+f);

break;

}

}

if(f==NFig)

std::cout<<'.';

if(i%3==2)

std::cout<<' ';

if(i%9==8)

std::cout<<std::endl;

}

}

void Solve(int x,int mb,int z)

{

while(z!=0)

{

if((x&z)==0)

break;

z>>=1;

mb--;

}

if(mb<0)

{

NSolve++;

Print();

return;

}

for(int f=0;f<NFig;f++)

if(FState[f]<0)

{

Fig &F=Figs[f];

int r1=F.Ind[mb],r2=F.Ind[mb+1];

int \*aa=F.Arr+r1;

for(int u=r1;u<r2;u++)

{

int s=\*aa++;

if((x&s)==0)

{

FState[f]=u;

Solve(x|s,mb,z);

}

}

FState[f]=-1;

}

NBack++;

}

int main()

{

Figs[0].Init(4,LL);

for(int i=1; i<NFig; i++)

{

Figs[i].Init(Figs0[i]);

FState[i]=-1;

}

int ns=0;

for(int fp=0; fp<Figs[0].NF; fp++)

{

FState[0]=fp;

Solve(Figs[0].Arr[fp],26,1<<26);

if(fp==1)

ns=NSolve;

}

std::cout<<"Решений="<<NSolve+ns<<std::endl;

std::cout<<"Переборов="<<NBack<<std::endl;

return 0;

}

***Результат работы программы:***

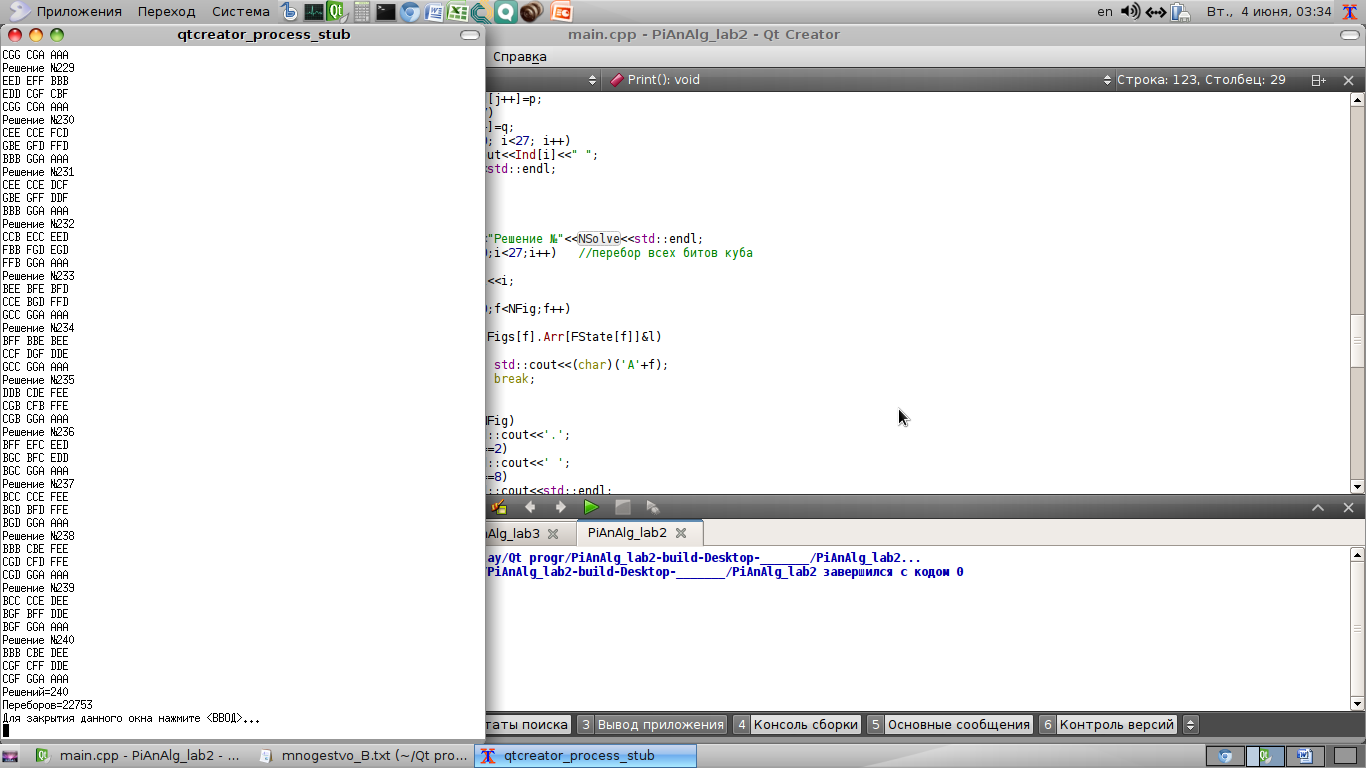
******

Рисунок 2 – результат работы программы.

Как видно из рисунка 2, мы получили 240 решений головоломки, что совпадает с теоретическими данными.

***Вывод:*** В данной лабораторной работе я познакомился с головоломкой «кубик Сома», написал алгоритм поиска ее решений. Полученный результат совпал с теоретическим.