**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 №3

по дисциплине «Анализ и проектирование алгоритмов»

Выполнил: Черницын Н. А.

Факультет: КТИ

Группа: 9301

Проверил: Казаков Б.Б.

#### Санкт-Петербург,

# 2013

# ***Формулировка задания:*** Профессор хочет экспериментально исследовать на прочность транзисторы. Он выбрал следующий способ проведения эксперимента: профессор намерен, перемещаясь по пожарной лестнице, сбрасывать транзисторы с высоты различных этажей. Таким образом, он планирует определить, при падении с какого минимального этажа транзистор разбивается. При этом профессору известно, что транзистор не выдерживает падения с последнего этажа, а падение с первого этажа не причиняет ему (транзистору) вреда.

Известно, что все транзисторы данной партии абсолютно одинаковы, и если транзистор разбивается при падении с некоторого этажа, то он разбивается и при падении с большей высоты. Разбившиеся транзисторы снова использовать нельзя, а оставшиеся целыми после падения могут использоваться повторно. Для того чтобы поднять оставшийся целым транзистор, профессору надо спуститься на первый этаж. Оказавшись на первом этаже, профессор может поднять все лежащие там транзисторы. Профессор хочет минимизировать суммарное расстояние, которое ему придется в худшем случае *подниматься* по лестнице. У профессора хорошее зрение, и он может с любого этажа определить, разбился транзистор или нет.

Изначально профессор находится на первом этаже, и у него имеется *m* транзисторов. В доме, где проводится эксперимент, *n* этажей, и все они имеют одинаковую высоту (её можно принять за единицу измерения расстояния).

***Теоретическая часть:***

Метод ветвей и границ (англ. branch and bound) — общий алгоритмический метод для нахождения оптимальных решений различных задач оптимизации, особенно дискретной и комбинаторной оптимизации. Метод характеризуется частичным целенаправленным перебором возможных вариантов. При этом решаемая задача последовательно ветвится, заменяясь более простыми, и путем анализа с помощью графа “дерево задач” отбрасываются заведомо непригодные варианты, чем облегчается дальнейший перебор.

Динамическим программированием называется метод, который позволяет решать «переборные» задачи, опираясь на уже решенные подзадачи меньшего размера.

Требования динамического программирования:

* Все решения подзадач должны запоминаться в таблице;
* Существуют известные решения для задачи с малой размерностью;
* Существует способ выразить решение задачи через подзадачи меньшей размерностью.

***Алгоритм работы программы:***

Известно, что транзистор не выдерживает падения с последнего этажа, а падение с первого этажа не причиняет ему (транзистору) вреда.

В программе используется рекурсия.

Т.к. мы знаем, что с первого этажа транзистор не разобьется, сразу проверяем второй.

При входе в функцию make\_test (смотри листинг), счетчик становится равным количеству транзисторов, которое экспериментатор имеет с собой.

Выкидываем один транзистор и проверяем, разбился ли он.

Если транзистор разбился, то проверяем, находится ли этот этаж выше на один этаж, при падении с которого транзистор еще остается цел.

Если да, то мы нашли решение.

Иначе увеличиваем счетчик транзисторов на земле. Запоминаем новый верхний этаж, с которого не разбивается транзистор.

Пока не найдено решение, выполняем следующие шаги:

* Если транзисторов у экспериментатора не осталось, то идем вниз и собираем целые транзисторы с земли. Инициализация счетчиков.
* Если количество целых транзисторов больше 2 и максимальный этаж, на который можно подняться за следующий шаг, не равен 0, то мы поднимаемся на следующий этаж (номер следующего этажа равен расстоянию от текущего этажа до верхнего этажа, с которого бьется транзистор, деленное на 2).
* Передаем расстояние, на которое мы поднялись, в функцию (рекурсивно) – это и будет количество этажей, на которые мы можем подняться.
* Иначе: мы поднимаемся на один этаж.

***Листинг:***

* ***Main.cpp***

#include <iostream>

#include "professor.h"

int main()

{

int Nfloor, Ntransistor, Ncrash;

std::cout<<"Введите количество этажей:";

std::cin>>Nfloor;

std::cout<<"Введите количество транзисторов, доступных для испытаний:";

std::cin>>Ntransistor;

std::cout<<"Введите этаж, с которого считать, что транзистор разбивается:";

std::cin>>Ncrash;

Professor pr(Ntransistor,Nfloor,Ncrash);

pr.make\_test(2,Nfloor);

pr.print();

}

* ***Professor.cpp***

#include "professor.h"

Professor::Professor(int transistor, int floor, int crash\_floor)

{

way\_length=0;

new\_transistors=transistor;

discarded\_transistors=0;

crash=crash\_floor;

low=1;

high=floor;

}

int Professor::make\_test(int floor, int divider)

{

//поднять транзисторы

int ret=0;

//print();

new\_transistors--;

way\_length+=divider;

std::cout<<"------------------------------------------------------------------------"<<std::endl;

std::cout<<"Этаж:"<<floor<<std::endl;

if (!(floor<crash)) //транзистор бъется

{

std::cout<<"Транзистор разбит!!!"<<std::endl;

high=floor;

if (low+1==floor)

{

low=floor;

return 1;

}

way\_length+=divider;

return 0;

}

low=floor;

discarded\_transistors++;

for (int i=2; !ret; i\*=2)

{

if (!(new\_transistors))

{

std::cout<<"Спускаемся взниз и забираем все неразбитые транзисторы. Текуший этаж:"<<floor<<std::endl;

new\_transistors=discarded\_transistors;

discarded\_transistors=0;

way\_length+=(floor\*2);

}

if (!(new\_transistors+discarded\_transistors<2) && (int)(divider/i)) //транзисторов больше двух

ret=make\_test(floor+divider/i,divider/i);

else //транзисторов меньше двух

ret=make\_test(floor+1,2);

}

return 1;

}

void Professor::print()

{

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

std::cout<<"Пройденный путь:"<<way\_length<<std::endl;

std::cout<<"Минимальная высота разбития транзистора:"<<low<<std::endl;

std::cout<<"Осталось неразбитых транзисторов:"<<new\_transistors+discarded\_transistors<<std::endl;

}

* ***Professor.h***

#ifndef PROFESSOR\_H

#define PROFESSOR\_H

#include <iostream>

class Professor

{

public:

Professor(int transistor, int floor, int crash\_floor);

int make\_test(int floor, int divider);

void print();

private:

int way\_length;

int new\_transistors, discarded\_transistors, crash;

int low,high;

};

#endif // PROFESSOR\_H

***Результат работы программы:***

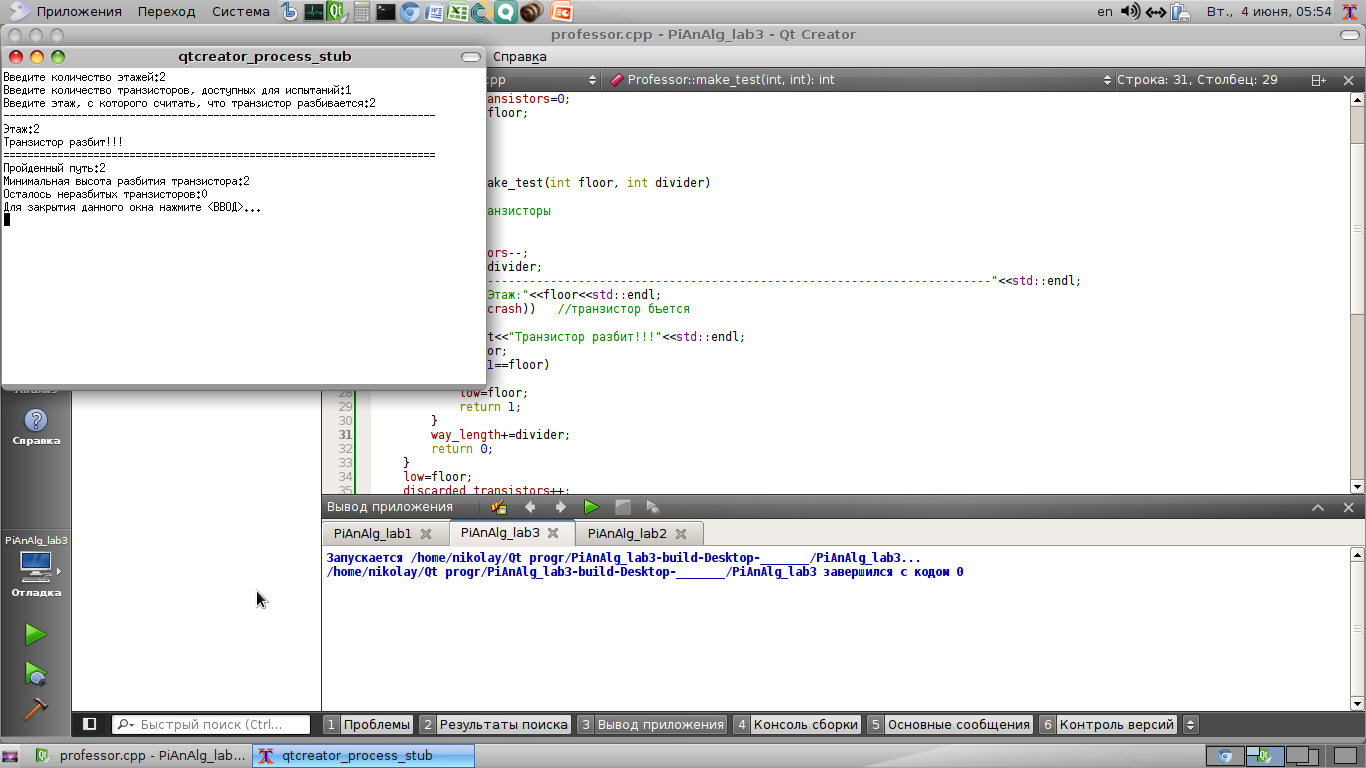
******

Рисунок 1 – результат работы программы

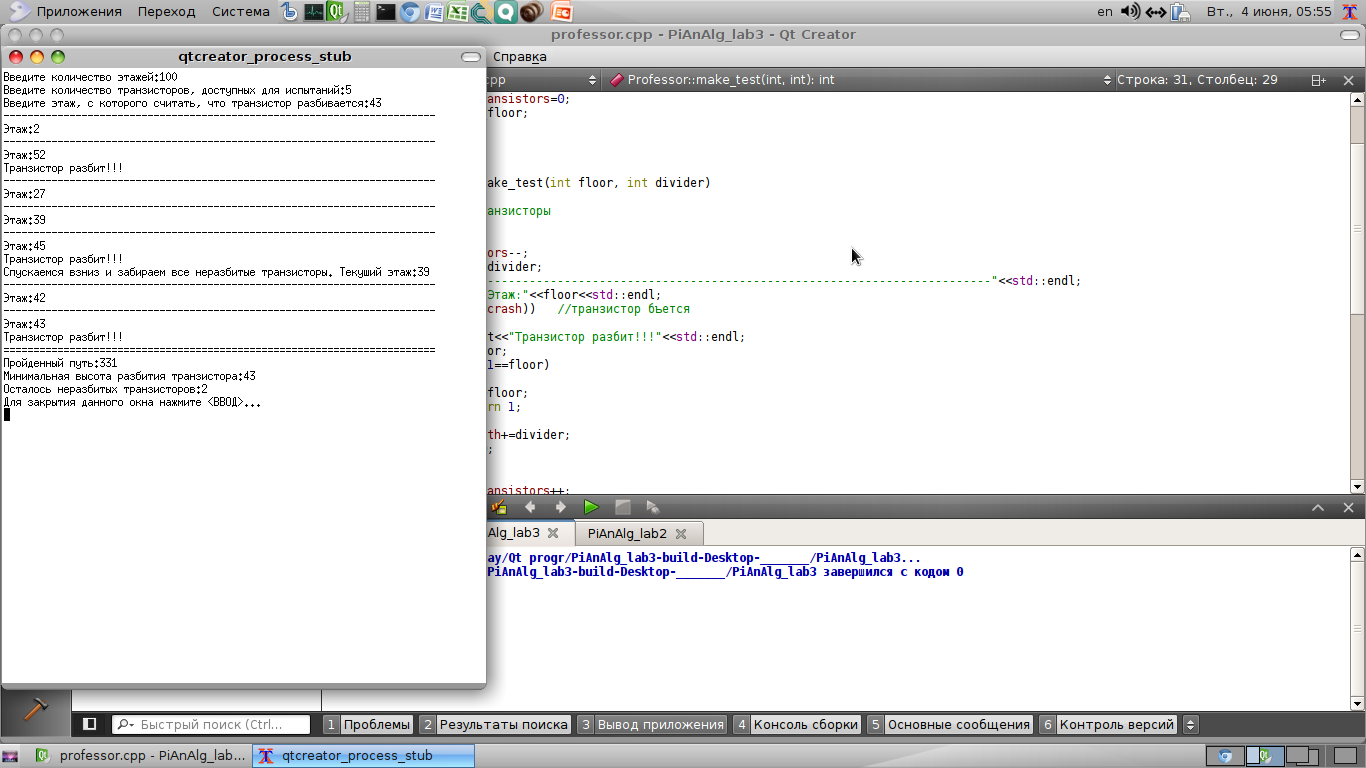


Рисунок 2 – результат работы программы

***Вывод:*** В данной лабораторной работе я познакомился с методом ветвей и границ и узнал о возможностях динамического программирования. На основе двух этих понятий я реализовал алгоритм поиска нахождения минимального этажа, с которого разобьется транзистор.