**Содержание**

Введение

1. Задание на курсовую работу
2. Разработка алгоритма решения задачи
3. Разработка программы
4. Тестирование и отладка программы
5. Документирование программы

Заключение

Литература

Приложение

**Введение**

Целью данной курсовой работы является получение навыков разработки, отладки, тестирования и документирования программ на языке высокого уровня при решении на ЭВМ прикладной инженерной задачи.

В ходе выполнения курсовой работы решены следующие задачи:

Разработан алгоритм синтеза пленочного резистора по заданным параметрам исходного резистора.

Разработана программа реализации данного алгоритма на языке С++.

Выполнена отладка и тестирование программы.

Составлена документация на программу и инструкция пользователя программой.

Программа оттранслирована и скомпонована в среде MVS-6 в виде консольного приложения.

Программа предназначена для работы под управлением операционной системы WINDOWS XP или ее более ранних версий.

При работе программы используется стандартное оборудование компьютера: процессор типа PENTIUM, жесткий диск, монитор, клавиатура и мышь.

* 1. **Задание на курсовую работу**

Прикладной инженерной задачей в данной курсовой работе является синтез интегральных резисторов микросхем. Резисторы микросхемы представляют собой полоски тонкой пленки в форме прямоугольника или меандра, как показано на рис. 1.

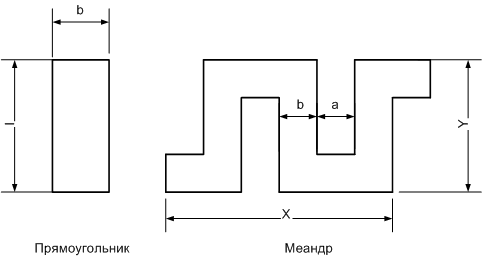


Рис. 1 Форма пленочного резистора

Расчет резистора заключается в определении его размеров – длины  и  ширины в случае прямоугольных пленок и ширины , расстояния  между полосками, размеров  и , а также числа звеньев  в случае пленок в форме меандра.

Исходные данные представлены в виде двух групп – данные с переменными и постоянными значениями.

Таблица 1. Исходные данные с переменными значениями

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Обозначения и ед | , кОм | , кОм/кв | , мВт | , мВт/мм2 |
| Наименование | Номинальное сопротивление | Сопротивление квадрата резистивной пленки | Мощность рассеяния резистора | Максимальная удельная мощность рассеяния резистивной пленки |
| Диапазон значение | 0.05..10000 | 0.01..50 | 5..50 | 10..30 |

Таблица 2. Исходные данные с постоянными значениями

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Обозначения и единицы | , мм | , мм | , мм | , мм |  | , мм | , мм |
| Наименование | Минимально допустимая ширина и длина резистора | | Погрешность воспроизведения ширины и длины резистора | | Допустимая погрешность коэффициента формы | Минимальное расстояние между полосками резистора сложной формы | Шаг координатной сетки |
| Диапазон значение | 0.1 | 0.3 | 0.01 | 0.03 | 0.1 | 0.3 | 0.01 |

Расчет резисторов выполняется в следующей последовательности.

Определяется значение коэффициента формы

 (1)

Выбирается форма резистора

* + 1. При  – резистор прямоугольной формы,
    2. При  – резистор прямоугольной формы, у которого ,
    3. При  – резистор сложной формы типа меандр,
    4. При  – резистор не может быть сконструирован.

Связь между размерами резистора прямоугольной формы определяется соотношением

 (2)

В случае 1

, (3)

где

, (4)

. (5)

После определения  длина вычисляется по формуле (2). Полученные значения ширины и длины округляются с учетом шага координатной сетки по формуле

, (6)

где ,  – целая часть числа.

В случае 2

Определяется длина из выражения

, (7)

где

, (8)

. (9)

Ширина  определяется по формуле (2) и значения ширины и длины округляются по формуле (6).

В случае 3

Определяется ширина резистора по формулам (3)–(5). Определяется длина средней линии меандра из выражения

. (10)

Задается расстояние . Если , то .

Определяется шаг одного звена меандра

. (11)

Определяется по приближенной формуле число звеньев меандра

. (12)

Вычисляется длина и ширина меандра

, (13)

 (14)

Проверяется условие

. (15)

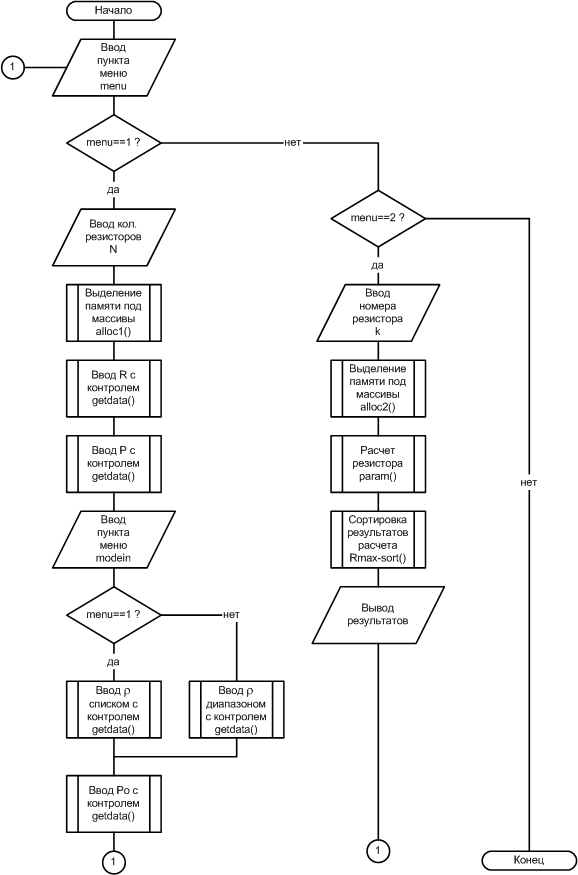
Если это условие выполняется, то расчет завершается. Иначе, увеличивается расстояние  на величину  и повторяются вычисления по формулам (11)–(14). Если при 10-ти кратном повторении соотношение (15) не выполняется, то расчет завершается с фиксированием факта невозможности синтеза резистора.

Рассмотренная методика расчета применяется к заданному количеству резисторов . Диапазон значений  и значение  для всех резисторов принимается одинаковыми.

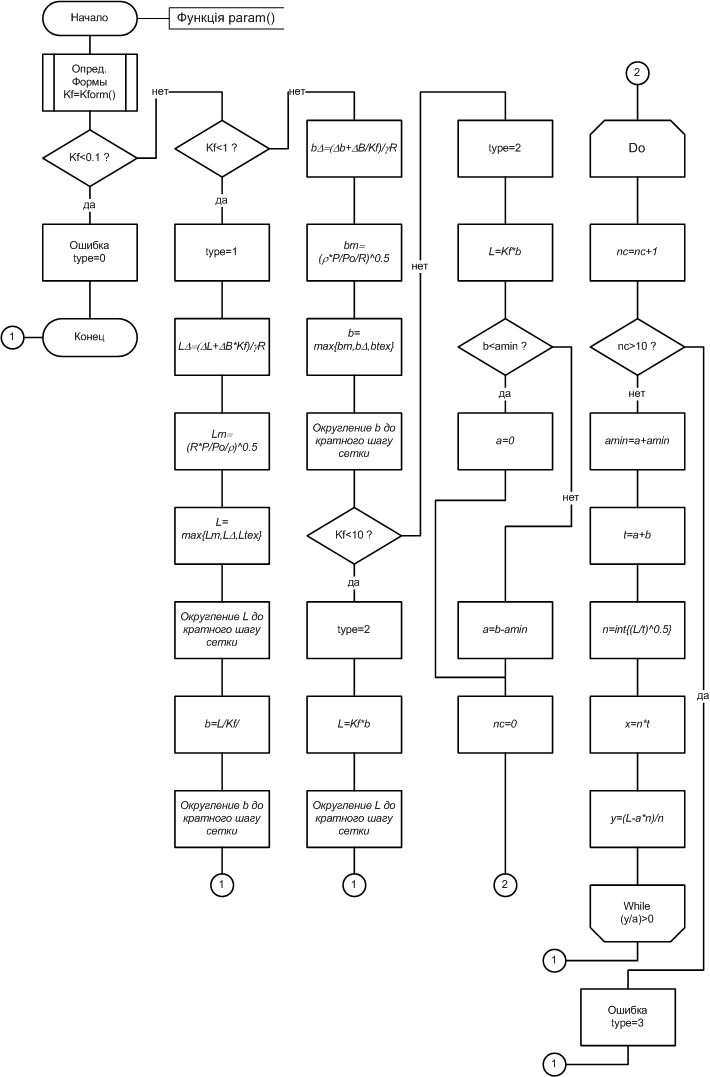
* 1. **Разработка алгоритма решения задачи**

Блок-схема головной программы

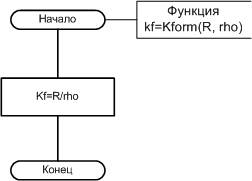
.



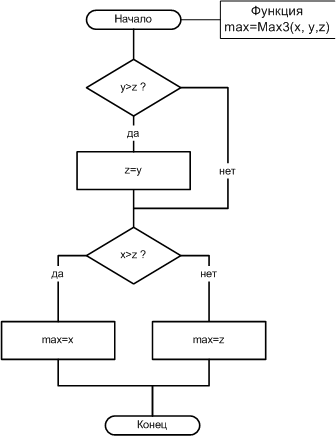
Блок-схема программы расчета параметров резистора



Блок-схема программы расчета коэффициента формы



Блок-схема программы расчета максимального из 3-х чисел



* 1. **Разработка программы**

При программировании использован аппарат функций, реализующих относительно самостоятельные участки алгоритма решения задачи. При составлении программы применялись как функции с аргументами и возвращаемыми значениями, так и функции, имеющие форму процедур без аргументов, в которых использовались глобальные данные программы.

При резервировании памяти использовались динамические массивы, указателям которых присваивались адреса памяти в ходе выполнения программы по количеству введенной информации. Для этой цели в языке С++ наиболее удобно использовать оператор

<указатель>= new <тип массива>[<количество элементов массива>];

При программировании меню удобнее всего использовать оператор switch (<условие>) {case <значение условия>: <исполняемый код> break; ….}

При организации циклов использованы три вида операторов:

for (i=<нач. знач>; i<<кон. знач>; i++) {<исполняемый код>};

do {<исполняемый код>} while (<условие повторения>);

while (<условие повторения>) {<исполняемый код>}.

* 1. **Тестирование и отладка программы**

При тестировании программы были рассчитаны все варианты синтеза пленочного резистора для нескольких исходных значений сопротивления, мощности рассеяния резистора и удельной мощности рассеяния пленки. Результаты тестирования, приведенные в приложении, показали корректность синтеза пленочного резистора.

* 1. **Документирование программы**

При запуске программы в среде WINDOWS XP в консольном приложении в окне программы появляется меню, состоящее из 3‑х пунктов:

* + 1. ввода данных с контролем их значений как визуально, так и по заданному допустимому диапазону,
    2. выполнения расчетов с отображением результатов,
    3. выхода из программы.

Если ввод данных не был произведен, то расчетная часть программы автоматически блокируется и предлагается либо ввести данные, либо выйти из программы.

При вводе данных предлагается ввести количество резисторов. После этого последовательно поступает на экран предложения ввести сопротивление резистора и его рассеиваемую мощность. Количество предложений формируется автоматически.

Затем предлагается указать тип ввода : по списку или по диапазону с заданным шагом. В первом случае предлагается ввести количество элементов списка, а затем поступают предложения ввода значений . Количество предложений контролируется автоматически. Во втором случае предлагается ввести начальное, конечное значение  и шаг его изменения.

После ввода  предлагается ввести значение . Все значения  и  являются одинаковыми для всех исходных резисторов.

На этом этап ввода заканчивается и выводится введенная информация для визуального контроля. При вводе данных автоматически контролируется их допустимый диапазон.

При расчете предлагается ввести номер резистора, для которого необходимо провести синтез пленки. Если номер резистора выходит за пределы их количества, то предлагается ввести новый корректный номер.

После ввода номера резистора осуществляется расчет параметров пленки с автоматическим выбором ее геометрии. Результаты расчета выводятся на экран в виде записей, отсортированных по возрастающей величине . Кроме того выводится максимальное значение .

Далее выводится исходное меню и работа с программой может быть повторена.

**Заключение**

1. Использование программирования на языке С++ позволяет реализовать сложные, многовариантные алгоритмы синтеза пленочных резисторов с автоматическим выбором геометрии пленки.
2. Данный язык программирования позволяет построить диалоговые программы с автоматическим управлением диалога и слежением за корректностью значений вводимых исходных данных.
3. Компилятор языка С++ строит эффективную по быстродействию программу и позволяет использовать динамическое распределение памяти, что дает возможность создать компактные по размерам программы.

**Литература**

* 1. С++ за 21 день.
  2. Программирование на С++. Учеб. пособие / Под ред. А.Д. Хомоненко. – С-Пб.: 1999.
  3. Освоение BORLAND C++ 4.5. Т1,2. К. Диалектика, 1996.

**Приложение**

 // подключение готовых библиотек

#define WIN32\_LEAN\_AND\_MEAN

#include <stdio.h>

#include <tchar.h>

#include <iostream>

#include <conio.h>

#include <iomanip>

#include <math.h>

using namespace std; // использование стандартного пространства имен

 // Не изменяемые расчетные параметры

double bt=0.1; // Техн. значение ширины пленки

double lt=0.3; // Техн. значение длины пленки

double Db=0.01; // Погрешность воспроизведения ширины

double Dl=0.03; // Погрешность воспроизведения длины

double gamma=0.1; // Допустимая погрешность коэф. формы

double amin=0.3; // Макс. расстояние между полосками пленки сложной формы (изменено)

double h=0.05; // Шаг сетки

 // Ограничения значений исходных данных

double Rdown=0.05; // Мин. значение сопротивления резистора

double Rup=10000; // Макс. значение сопротивления резистора

double rhodown=0.001; // Мин. значение удельного сопротивления пленки

double rhoup=1000; // Макс. значение удельного сопротивления пленки

double Pdown=5; // Мин. зсначение рассеиваемой мощности резистора

double Pup=50; // Макс. значение рассеиваемой мощности резистора

double Podown=10; // Мин. зсначение удельной рассеиваемой мощности пленки

double Poup=30; // Макс. зсначение удельной рассеиваемой мощности пленки

double R, rho, P, Po, b, l, a, t, x, y, kf, Sglob;

double RhoBeg, RhoEnd, RhoStp, RhoMdl;

int n, NResist, NRho;

int type, nc, menu, cntrl, modein, k;

double \*ArrR=0;

double \*ArrP=0;

double \*ArrRho=0;

double \*ArrKf=0;

double \*ArrB=0;

double \*ArrL=0;

double \*ArrA=0;

double \*ArrT=0;

double \*ArrX=0;

double \*ArrY=0;

double \*ArrS=0;

int \*ArrType=0;

int \*ArrN=0;

int \*index;

double \*ArrBuf;

 // для вычисление коэффициента формы double KForm (double R, double rho);

 // для нахождение максимального из трех чисел double Max3 (double x, double y, double z);

 // для нахождение миниимального из трех чисел double Min3 (double x, double y, double z);

 // для расчет параметров пленки void ParamR(void);

 // Ввод расчетных параметров с контролем их диапазона double getdata (const char \*partype, double pardown, double parup);

 // для сортировки массивов по возрастанию int sort(void);

 // Вывод результатов расчета пленки void OutResult(void);

 // Вывод расчетных параметров пленки void OutParam(void);

 // Освобождение памяти для массивов ArrB и ArrP void alloc1 (void);

 // Выделение и освобождение памяти для остальных динамических массивов void alloc2 (void);

int main()

{

int j;

cntrl=0;

double nbmax;

do

{

cout

<<»\tM E N U \n»<<endl

<<» [1] – VVOD PARAMETROV»<<endl

<<» [2] – RASCHET»<<endl

<<» [0] – VIHOD»

<<endl;

cin>>menu;

switch(menu)

{

case 1:

cout<<»\nKolichestvo Resistirov N=»;

cin>>NResist;

 // Выделение динамической памяти под массивы R и P

alloc1 ();

for (j=0; j<NResist; j++)

{

cout<<»\nResistor»<<setw(2)<<j+1<<»:»;

ArrR[j]=getdata («\nR(kOm) (0.05–10000)», Rdown, Rup);

ArrP[j]=getdata («P(mW) (5–50)», Pdown, Pup);

}

cout

<<»\nMetod zadaniya Diapozona Rho\n\n»

<<» [1] – SPISOK Rho (rho1, rho2,…)«<<endl

<<» [2] – DIAPAZON Rho (rho\_begin, rho\_end, rho\_step)«<<endl

<<endl;

cin>>modein;

switch(modein)

{

case 1: // Ввод числа элементов в списке сопротивлений пленки

cout<< «Razmer spiska NRho=»;

cin>>NRho;

break;

case 2: // Ввод диапазона и шага изменения сопротивлений пленки

RhoBeg=getdata («RhoBeg», rhodown, rhoup);

RhoEnd=getdata («RhoEnd», rhodown, rhoup);

RhoStp=getdata («RhoStp», rhodown, rhoup);

NRho=(int) ((RhoEnd-RhoBeg)/RhoStp)+1;

rho=RhoBeg-RhoStp;

break;

}

alloc2 (); // Выделение динам. памяти под массив сопротивления пленки rho

for (j=0; j<NRho; j++)

{

 // Ввод элементов массива значений rho с заданным шагом в диапазоне

if (modein==1) ArrRho[j]=getdata («rho», rhodown, rhoup);

 // Ввод элементов массива значений rho по списку

if((modein==2) | (modein==3))

{

rho+=RhoStp;

ArrRho[j]=rho;

}

}

Po=getdata («Po (mW/mm2) (10–30)», Podown, Poup); // Ввод значения удельной мощности Po

cntrl=1;

OutParam(); // Вывод исх. данных для визуального контроля

break;

case 2: // Рассчет геометрических параметров пленки

if(cntrl)

{ // Проверка наличия исх. данных

while (1==1)

{

cout<<»\nNomer Resistora (1. «<<NResist<<») k=»;

cin>>k; k –;

if (k>=0 && k<NResist) break;

}

R=ArrR[k];

P=ArrP[k];

for (j=0; j<NRho; j++)

{

rho=ArrRho[j];

ParamR();

ArrType[j]=type;

ArrKf[j]=kf;

ArrB[j]=b;

ArrL[j]=l;

ArrA[j]=a;

ArrX[j]=x;

ArrY[j]=y;

ArrT[j]=t;

ArrN[j]=n;

ArrS[j]+=b\*l;

}

nbmax=sort();

OutParam();

for (j=0; j<NRho; j++) ArrBuf[j]=ArrB[j];

nbmax=sort();

cout<<» MIN b="<<ArrB [index[0]]<< «mm»<<endl;

cout<<» MAX b="<<ArrB [index[NRho‑1]]<< «mm»<<endl<<endl;

for (j=0; j<NRho; j++) {

type=ArrType [index[j]];

rho=ArrRho [index[j]];

kf=ArrKf [index[j]];

b=ArrB [index[j]];

l=ArrL [index[j]];

a=ArrRho [index[j]];

x=ArrX [index[j]];

y=ArrX [index[j]];

t=ArrT [index[j]];

n=ArrN [index[j]];

OutResult();

}

cout<<endl;

}

else

cout<< «VVEDITE PARAMETRI»<<endl;

break;

case 0:

cout<<endl<<endl<<endl<< «Press ENTER to EXIT…»;

getch();

return 0;

}

}

while (1==1);

}

 // Вычисление коэффициента формы

 // Аргументы: R‑сопротивление резистора, rho‑сопротивление пленки

 // Вывод: коэффициент формы double KForm (double R, double rho) {return R/rho;}

 // Определение максимального из 3‑чисел x, y, z

 // Аргументы: числа x, y, z

 // Вывод: макс {x, y, z}

double Max3 (double x, double y, double z) {return x>(z=y>z? y:z)? x:z;}

double Min3 (double x, double y, double z) {return x<(z=y<z? y:z)? x:z;}

 // Вычисление геометрических размеров пленки void ParamR()

{

kf=KForm (R, rho);

if (kf<0.1) type=0; // недопустимое значение Kf

else if (kf<1)

{ // прямоуголиная пленка при kf<1 (l<b) ф. (7) – (9), (2)

type=1;

double lD=(Dl+Db\*kf)/gamma;

double lP=sqrt (R\*P/rho/Po);

l=((int) (Max3 (lt, lD, lP)/h)+1)\*h;

b=((int) (l/kf/h)+1)\*h;

}

else

{ // прямоуголиная пленка при 1<=Kf<=10 (l>b) ф. (2) – (6)

double bD=(Db+Dl)/kf;

double bP=sqrt (rho\*P/R/Po);

b=((int) (Max3 (bt, bD, bP)/h)+1)\*h;

if (kf<=10)

{

type=2;

l=((int) (kf\*b/h)+1)\*h;

}

else

{ // пленка типа меандр при Kf>10 ф. (10) – (15)

type=3;

l=kf\*b;

if (b<amin) a=0;

else a=b-amin;

nc=0;

do

{

nc++;

if (nc>10)

{

type=4;

break;

}

a+=amin;

t=a+b;

n=(int) sqrt (l/t);

x=n\*t;

y=(l-a\*n)/n;

}

while (y/a>10);

}

}

}

 // Ввод исходных данных с проверкой допустимого их диапазона значений

 // Аргументы:

 // строка, содержащая название параметра,

 // нижняя граница допустимого значения,

 // верхняя граница допустимого значения

 // Выход: значение параметра double getdata (const char \*partype, double pardown, double parup)

{

double parval;

while (1==1)

{

cout<<partype<< "=»; cin>>parval;

if (parval<pardown || parval>parup)

cout<< «non corect DIAPASON»<<endl;

else break;

}

return parval;

}

 // Сортировка индекса массива по возрастанию методом пузырька

 // Результат сортировки сохраняется в индексном массиве,

 // элементы которого по порядку, начиная с младшего, содержат номера

 // элементов исходного массива в порядке их возрастания, т.е.

 // первый(нулевой) элемент индексного массива содержит номер

 // минимального элемента исходного массива, а последний элемент индексного

 // массива содержит номер максимального элемента исходного массива

 // На выход подается номер максимального элемента исходного массива

int sort()

{

int j, jtmp;

int flag;

double btmp;

double \*buf;

buf=new double[NRho];

for (j=0; j<NRho; j++)

{

index[j]=j;

buf[j]=ArrBuf[j];

}

do

{

flag=0;

for (j=1; j<NRho; j++)

if (buf[j‑1]>buf[j])

{

jtmp=index[j];

index[j]=index [j‑1];

index [j‑1]=jtmp;

btmp=buf[j];

buf[j]=buf [j‑1];

buf [j‑1]=btmp;

flag=1;

}

}

while(flag);

return index[NRho];

}

 // Вывод введенных исходных данных для визуального контроля

void OutParam()

{

int i, j;

cout<<endl;

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

{

cout

<<» R»<<i+1<< "="<<ArrR[i]<< «kOm»

<<» P="<<ArrP[i]<< «mW»

<<» Po="<<Po<< «mW/mm2»;

if (modein==1)

{

cout<<» rho={»;

for (j=0; j<NRho‑1; j++)

cout<<ArrRho[j]<<»;»;

cout<<ArrRho [NRho‑1]<<») kOm»<<endl;

}

if((modein==2)|(modein==3))

cout

<<» rho=(«<<RhoBeg

<<»; «<<RhoBeg+RhoStp

<<». «<<RhoEnd<<») kOm/mm2»<<endl;

}

cout<<endl;

}

 // Вывод результатов расчета пленочного резистора void OutResult()

{

switch(type)

{

case 0: // Ошибка Kf<0.1

cout<<«ERROR «<< «Kform=»<<kf<<endl;

break;

case 1:

case 2: // прямоугольная пленка 0.1<=Kf<=10

cout<<setw(9)

<<» Rectangle»

<<setprecision(3)

<<» b="<<setw(4)<<b<< «mm»

<<» l="<<setw(4)<<l<< «mm»

<<» S="<<setw(4)<<b\*l<< «mm2»

<<» Kf="<<setw(4)<<kf

<<»\tRho="<<setw(6)<<rho<< «kOm/mm2» //rho

<<endl;

break;

case 3:// пленка типа меандр Kf>10

cout<<setw(6)

<<» Meandr»

<<setprecision(3)

<<» b="<<setw(4)<<b<< «mm»

<<» a="<<setw(4)<<a<< «mm»

<<» t="<<setw(4)<<t<< «mm»

<<» x="<<setw(4)<<x<< «mm»

<<» y="<<setw(4)<<y<< «mm»

<<» n="<<setw(2)<<n

<<» Kf="<<setw(4)<<kf

<<» Rho="<<setw(5)<<rho

<<endl;

break;

case 4:

cout<<«ERROR «<< «nc=»<<nc<<endl;

break;

}

}

 // Выделение памяти для динамических массивов значений

 // сопротивления R и рассеиваемой мощности P void alloc1 ()

{

if (ArrR!=NULL) delete ArrR;

if (ArrP!=NULL) delete ArrP;

ArrR=new double[NResist];

ArrP=new double[NResist];

}

 // Выделение памяти для динамических массивов значений

 // сопротивления пленки, коэффициентов формы и

 // рассчитанных геометричесих параметров пленочного резистора void alloc2 ()

{

if (ArrRho!=NULL) delete ArrRho;

if (ArrKf!=NULL) delete ArrKf;

if (ArrB!=NULL) delete ArrB;

if (ArrL!=NULL) delete ArrL;

if (ArrA!=NULL) delete ArrA;

if (ArrX!=NULL) delete ArrX;

if (ArrY!=NULL) delete ArrY;

if (ArrT!=NULL) delete ArrT;

if (ArrN!=NULL) delete ArrN;

if (index!=NULL) delete index;

if (ArrBuf!=NULL) delete ArrBuf;

if (ArrType!=NULL) delete ArrType;

ArrRho=new double[NRho];

ArrKf=new double[NRho];

ArrB=new double[NRho];

ArrL=new double[NRho];

ArrA=new double[NRho];

ArrX=new double[NRho];

ArrY=new double[NRho];

ArrT=new double[NRho];

ArrN=new int[NRho];

ArrS=new double[NRho];

ArrType=new int[NRho];

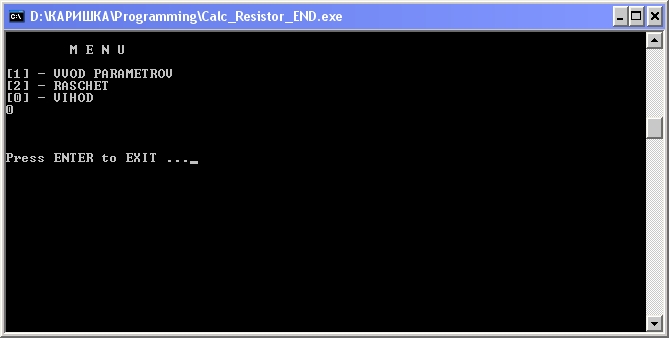
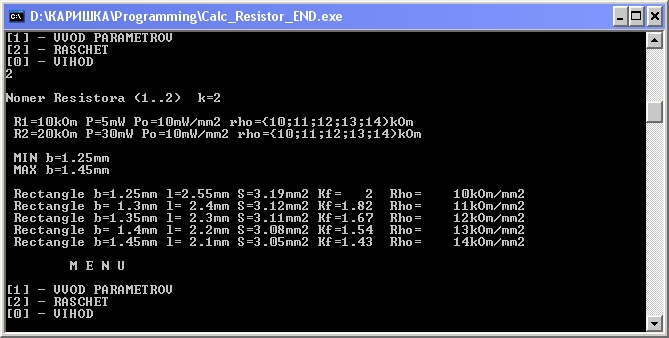
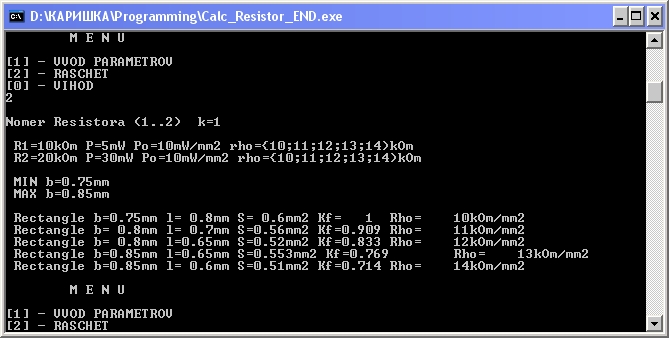
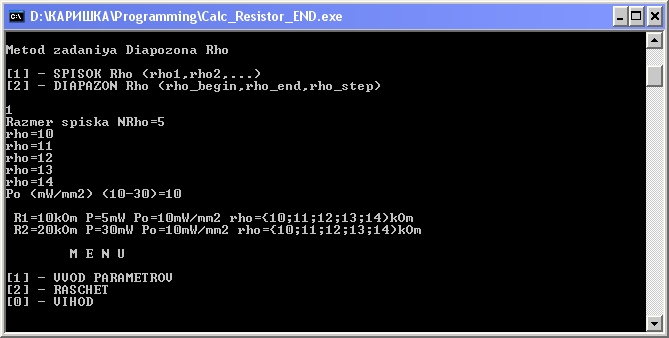
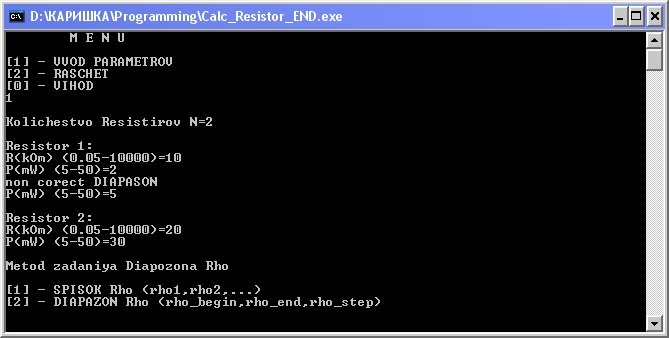
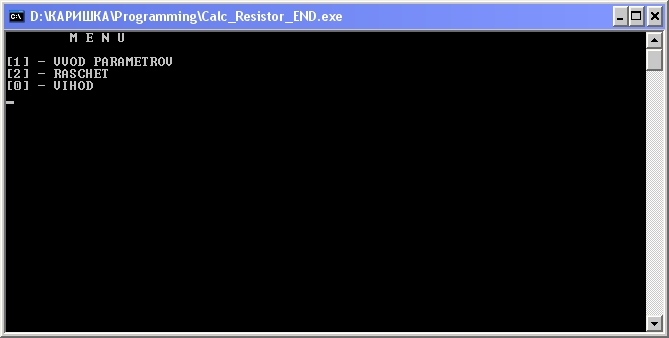
index=new int[NRho];

ArrBuf=new double[NRho];

for (int j=0; j<NRho; j++) ArrS[j]=0;

}

**Результаты тестирования программы**



Блок-схема программы контроля вводимых данных

**Начало**

**Ввод значения переменной – “parval” с контролем диапозона**

**Parval < parDown || > parUp**

**да**

**нет**

**Non correct DIAPOZON**

**Конец**

**Функция “getdata”**