

Работа над ошибками 1

0) Прографируйте проще! (Принцип KISS)

1) Аккуратно форматируйте код

2) Внимательно изучайте условие задачи

3) Не меняйте исходные данные

4) Не делайте лишнего

Задача относится к классу переборных задач (когда дано множество точек и необходимо построить множество **пар, **троек**, **четверок** и т.д.)**

Эти пары, тройки, четверки используются для вычисления расстояний, поэтому порядок элементов не важен и

пары (a,b) и (b,a) – это одна пара.

Строить выборки нужно рационально.

sqrt и **pow** в задачах не нужны – это лишние вычисления!

Вместо поиска максимальной длины проще искать квадрат максимальной длины.

Возведение вещественного в степень часто реализуется через **exp** и **ln**.

Для вычисления квадрата достаточно умножения.

Никогда не сравнивайте числа с
плавающей точкой на точное
равенство!

Вместо

if (a == b) ...

Пишите:

if (fabs(a-b) <= 1.0E-15) ...

Использование динамической памяти в этих задачах не предполагалось. Но если взяли память - освобождайте!

В некоторых работах каждая задача использовала **два цикла: в первом искался максимум, во втором осуществлялся вывод (в соотв. с найденным максимумом)...**

Площадь треугольника:

$$S=0.5*\text{abs}((x_1-x_3)(y_2-y_3)-(x_2-x_3)(y_1-y_3))$$

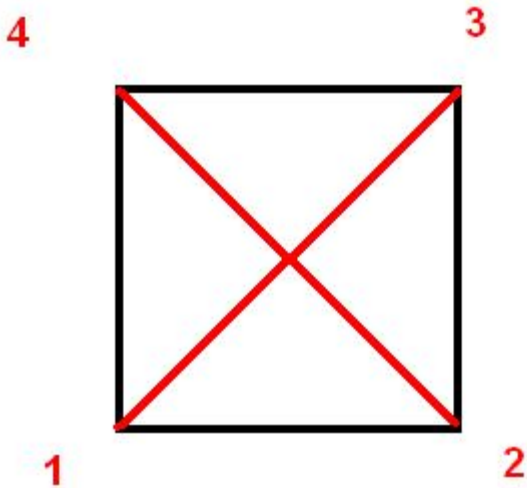
3 умножения, 5 вычитаний и модуль!

Формула Герона значительно затратнее! Не говоря уже о...

Если точки лежат на прямой – $S=0$

И самое главное...

Да, 4 точки образуют квадрат, если
равны между собой длины сторон и
равны диагонали...

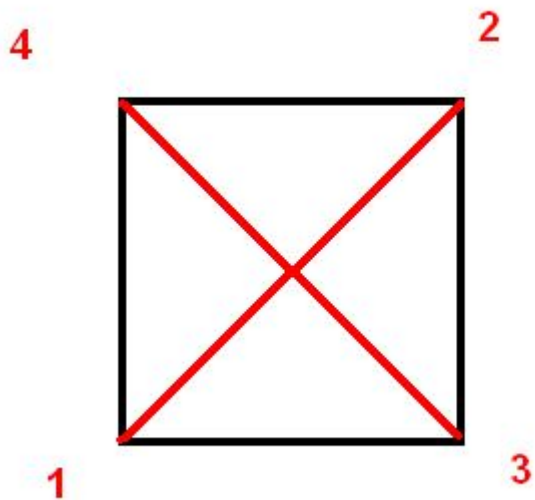


$$(1-2) = (2-3) = (3-4) = (1-4)$$

И

$$(1-3) = (2-4)$$

А если вершины перенумерованы по-другому, например так:



$$(1-3) = (2-3) = (2-4) = (1-4)$$

И

$$(1-2) = (3-4)$$

```

int isSqr(double *X, double *Y)
{
    int i,j,k,s1,s2,s3; double d,p,q,D[6];
    k=0;
    for (i=0; i<=2; i++)
        for (j=i+1; j<=3; j++)
            { p=X[i]-X[j];
              q=Y[i]-Y[j];
              d=p*p+q*q;
              D[k++]=d; }
    d=D[0];
    s1=1;
    s2=0;
    s3=0;
    for (i=1; i<6; i++)
        { if (fabs(d-D[i]) <=1.0E-15) s1++;
          if (fabs(2*d-D[i]) <=1.0E-15) s2++;
          if (fabs(d-2*D[i]) <=1.0E-15) s3++; }
    if ((s1==4 && s2==2) || (s2==2 && s3==4)) return 1;
    return 0;
}

```

Задача 2

Построение массива уникальных

Момент истины:

Не делать лишних сравнений

```
int main()
{
    int Arr[10] = {5, 2, 6, 5, 8, 6, 3, 3, 6, 4}, Arr_u[10];
    int p=10, i, j,n,k;
    Arr_u[0]=Arr[0];
    k=0;
    for (i=1; i<p; i++)
    {
        n=0;
        for (j=0; j<=k; j++) // сравнение с уникальными
            if (Arr[i]==Arr_u[j]) { n=1; break; }
        if (n == 0) Arr_u[++k]=Arr[i];
    }
    for (i=0; i<=k; i++) printf ("%d ", Arr_u[i]);
}
```

Задача 3

Слияние отсортированных массивов

Момент истины:

**Слияние, а не
объединение+сортировка**

```
int A[7]={1,3,4,5,6,8,12};
```

```
int B[6]={-1,0,7,8,8,20};
```

```
int R[13];
```

```
int i,j,k,p;
```

```
int na=7;
```

```
int nb=6;
```

```
i=j=k=0;
```

```
while (1)
```

```
{
```

```
    if (i > na) { for (p=j; p<nb; p++) R[k++]=B[p]; break; }
```

```
    if (j > nb) { for (p=i; p<na; p++) R[k++]=A[p]; break; }
```

```
    if (A[i]>B[j])
```

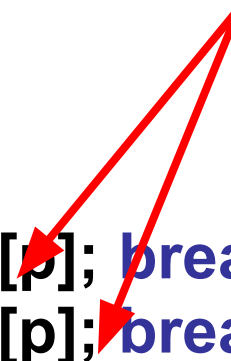
```
        R[k++]=B[j++];
```

```
    else
```

```
        R[k++]=A[i++];
```

```
}
```

К какому оператору относятся break ?



Зелёный зал

```
double squareArea( double a )  
{   return pow( a, 2 ); }
```



```
scanf("%i", &n);  
int x[n], y[n];
```

```
for (int i = 0; i < count; i++) {  
    if((FirstArr [i] == FinalArr [i]) || g)  
        g = true; // зачем крутить дальше?  
}
```

```
int n=0;
int a[8]={1,2,1,3,3,2,2,3};
int arr[n];
for (int i=1;i<=n;i++)
{
    if (a[i+1]!=a[i])
    {
        arr[i]=a[i];
        n++;
    }
}
```

```
number_list.sort() // А если в массиве 10 млн. эл-тов?  
unique_number_list = [number_list[0]]  
for i in range(1, len(number_list)):  
    if number_list[i] != number_list[i - 1]:  
        unique_number_list.append(number_list[i])  
print(number_list, unique_number_list, sep=' ---> ')
```

```
int n = 6, mass[n] = {-1, 1, 2, 2, 2, 0};
```

```
for(int i = 0; i < n; ++i)
    for(int j = i + 1; j < n;){
        if(mass[i] == mass[j]){
            for(int l = j; l < n; ++l)
                mass[l] = mass[l+1];
            n--;
        }
        else j++;
    }
```

```
#include<conio.h>
#include<math.h>
#include<stdlib.h>
#include<stdlib.h>
#include<time.h>
#include<locale.h>
```

```
bool isAnElementHere(int* arr, int element){
    for(int i=0; i<sizeof(arr)/sizeof(int); i++)
        if(arr[i]==element) return true;
    return false;
}
```

```
for i in range(len(arr)):
    if arr[i] not in res:
        res.append(arr[i])
```

```
while(i<4 && j<4) // что такое 4???
```

```
{
```

```
    if(x[i]<y[j]) c[v++]=x[i++];
```

```
    else c[v++]=y[j++];
```

```
}
```


double

**length=leng(X[i],X[j],Y[i],Y[j]);//находим
длину по координатам**

```
for(j=0; j< NewLeng ; j++)  
{  
    if(X[i] == X[j]) break;  
}
```

```
/* Если не одно значения массиве в  
диап [0...j] не совпало с i, то текущее  
значение записываем как уникальное  
*/
```

```
if (j==NewLeng )  
    X[NewLeng++] = X[i];
```

```
// чему равно значение параметра  
цикла?
```

**//задача 3 объединить два массива и
отсортировать по возрастанию**

Откуда условие?

```
C:\Users\Bob\AppData\Local\Temp\tc\n1.cpp - Notepad++
Файл Правка Поиск Вид Кодировки Синтаксисы Опции Инструменты Макросы Запуск Плагины Вкладки ?
n1.cpp x
6 float l;
7 l=(float)((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2));
8 return l;
9 }
10
11 int main() {
12
13     int X[]={2,2,4,4,6,6,8};
14     int Y[]={6,4,4,6,2,6,4};
15     int l1=0, l2=0, t1=0, t2=0, t3=0, k1=0, k2=0, k3=0, k4=0;
16     float maxL=0, maxS=0, maxKS=0;
17
18     for (int i=0; i<sizeof(X)/sizeof(int); i++) {
19         for (int j=0; j<sizeof(X)/sizeof(int); j++) {
20             if (sqrt(sum(X[i], X[j], Y[i], Y[j]))>maxL) {
21                 maxL=sqrt(sum(X[i], X[j], Y[i], Y[j]));
22                 l1=i;
23                 l2=j;
24             }
25             for (int f=0; f<sizeof(X)/sizeof(int); f++) {
26                 if (0.5*fabs( (X[i]-X[f])*(Y[j]-Y[f])-(Y[j]-Y[f])*(X[j]-X[f]))>maxS) {
27                     maxS=0.5*fabs( (X[i]-X[f])*(Y[j]-Y[f])-(Y[j]-Y[f])*(X[j]-X[f]));
28                     t1=i;
29                     t2=j;
30                     t3=f;
31                 }
32                 for (int m=0; m<sizeof(X)/sizeof(int); m++) {
33                     if ( (sum(X[i], X[j], Y[i], Y[j])==sum(X[f], X[j], Y[f], Y[j])) && (sum(X[i], X[f], Y[i], Y[f])==sum(X[m], X[j], Y[m], Y[j])) && (sum(X[i], X[j], Y[i], Y[j])*sum(X[i], X[j], Y[i], Y[j])>maxKS) && (i!=f) ) {
34                         maxKS=sum(X[i], X[j], Y[i], Y[j]);
35                         k1=i;
36                         k2=j;
37                         k3=f;
38                         k4=m;
39                     }
40                 }
41             }
42         }
43     }
44     printf( "l=%f n1=%i n2=%i\n", maxL, l1, l2);
45     printf( "s=%f n1=%i n2=%i n3=%i\n", maxS, t1, t2, t3);
46     printf( "ks=%f n1=%i n2=%i n3=%i n4=%i\n", maxKS, k1, k2, k3, k4);
47     getch();
48 }
49
C++ source file length : 1 732 lines : 49 Ln : 33 Col : 207 Pos : 1 256 Windows (CR LF) ANSI INS
```

Длина строки 200 симв!

```
a=0;
for (int j=0;j<n;j++)
{
    if (a1[i]==a2[j]) break;
    a++;
}
if (a==n){
```

```
s=int(s*100 + .5); //округление числа  
    (например 1.9999 в 2)
```

```
s=s/100;
```

```
if(s==maxs){
```

```
int *numb;  
numb=(int*)malloc(1*sizeof(int)); // что в numb?
```

```
int i, j;  
for (i=0; i<l; i++){
```

```
// Проверяется, есть ли это число в numb
```

```
for (j=0; j<i;j++)  
    if (str[i] == numb[j]) break;
```

```
int uniquen(int num, int* arr, int len)
{
    for(int i=0; i<len; i++)
    {
        if(arr[i]==num) return 1;
    }
    return 0;
} // Проще проверить на месте !!!
```



```
int arr[]={1, 1, 1, 3, 4, 10, 0, 12, 3, 3, 8};  
len=sizeof(arr)/sizeof(int);  
bool *buf=(bool*) calloc (m=max(arr, len),  
    sizeof(bool));
```

// создается огромный булевский массив

**// проверка элемента на уникальность,
если элемент больше предыдущего и
меньше следующего, то элемент
массива уникальный**

```
for (int i = 0; i < 8; i++)  
{  
    for (int j = 0; i < 8; j++) // ???  
    {
```

```
int main()
{
    int arr[] = {1,2,1,3,3,2,2,3};
    int n,m=0;
    int res[ ]={};
    n = sizeof(arr)/sizeof(int);
    int i,j;
    for (i=0; i<n; i++)
    {
        j=0;
        while (arr[i]!=res[j]&& j<=m)
            j++;
        if(j>m)
        {
            res[m] = arr[i];
            m++;
        }
    }
    for (i=0;i<m; i++)
        printf("%d ", res[i]);
}
```

```
int *newmass=(int *)calloc(8, sizeof(int *));
```

Логика такая: изначальный массив по возрастанию отсортировать, затем сравнивая элементы отсортированного массива занулить не уникальные элементы, и вуаля.

Логика порочна: 1) как быть, если в массиве есть нули 2) зачем менять исходный массив? Это плохая практика. Не говоря уже о том, что сортировка – дорогая операция.

```
pl=((x[j]-x[i])*(y[k]-y[i])-(x[k]-x[i])*(y[j]-y[i]))/(float)2;
```

```
if (pl<0) pl*=-1;
```

```
pl=fabs(((x[j]-x[i])*(y[k]-y[i])-(x[k]-x[i])*(y[j]-y[i]))/2.0);
```

```
for ( int i = 1, a = 1; i < x; i++){  
    for ( int j = 1; j <= i; j++){  
        if ( simple_massive[i] ==  
            simple_massive[i-j] ){  
            goto fall;  
        }  
    }  
}
```