Convex Hull - Graham Scan Algorithm

Run on IDE (https://ide.codingblocks.com/#/s/6525)

```
#include <iostream>
#include <stack>
#include <stdlib.h>
using namespace std;

struct Point
{
    int x, y;
};


Point p0;

Point nextToTop(stack<Point> &S)
{
    Point p = S.top();
    S.pop();
    Point res = S.top();
    S.push(p);
    return res;
}

void swap(Point &p1, Point &p2)
{
    Point temp = p1;
    p1 = p2;
    p2 = temp;
}

int distSq(Point p1, Point p2)
{
    return (p1.x - p2.x)*(p1.x - p2.x) +
          (p1.y - p2.y)*(p1.y - p2.y);
}

int orientation(Point p, Point q, Point r)
{
    int val = (q.y - p.y) * (r.x - q.x) -
              (q.x - p.x) * (r.y - q.y);

    if (val == 0) return 0;
    return (val > 0)? 1: 2;
}

int compare(const void *vp1, const void *vp2)
{
   Point *p1 = (Point *)vp1;
```

```
   Point *p2 = (Point *)vp2;

   int o = orientation(p0, *p1, *p2);
   if (o == 0)
     return (distSq(p0, *p2) >= distSq(p0, *p1))? -1 : 1;

   return (o == 2)? -1: 1;
}

void convexHull(Point points[], int n)
{
   int ymin = points[0].y, min = 0,m=1;
   for (int i = 1; i < n; i++)
   {
     int y = points[i].y;

     if ((y < ymin) || (ymin == y &&
         points[i].x < points[min].x))
        ymin = points[i].y, min = i;
   }

   swap(points[0], points[min]);

   p0 = points[0];
   qsort(&points[1], n-1, sizeof(Point), compare);

   for (int i=1; i<n; i++)
   {

       while (i < n-1 && orientation(p0, points[i],
                                     points[i+1]) == 0)
          i++;
       points[m] = points[i];
       m++;
   }


   if (m < 3) return;

   stack<Point> S;
   S.push(points[0]);
   S.push(points[1]);
   S.push(points[2]);

   for (int i = 3; i < m; i++)
   {

      while (orientation(nextToTop(S), S.top(), points[i]) != 2)
         S.pop();
      S.push(points[i]);
   }
```

```
    while (!S.empty())
    {
        Point p = S.top();
        cout << "(" << p.x << ", " << p.y <<")" << endl;
        S.pop();
    }
}

int main()
{
    Point points[] = {{0, 3}, {1, 1}, {2, 2}, {4, 4},
                      {0, 0}, {1, 2}, {3, 1}, {3, 3}};
    int n = sizeof(points)/sizeof(points[0]);
    convexHull(points, n);
    return 0;
}
```