# 程 序 设 计 大 作 业 报 告

作业名称：城市天际线

学　　院：材料学院

班　　级：04102301

学生姓名：白昊明

学　　号：2023301350

西北工业大学

2023年12月2日

**摘要**

本次大作业选择的是上的ACM模拟试题，题目的主要要求是根据给出的矩形建筑物的上沿，输出城市的天际线的顶点向量。本次作业使用了C++风格的文件流操作，代码更加简洁易懂，并且融入了面向对象编程的思想。本次作业基本完成既定目标，实现了预期功能，但是对数据执行了三次循环，执行效率尚有待提高。本次作业思路清晰明确，运用了高级工具，运用了C++特有的功能和特性。

# 目录

# 1 摘要

## 1.1 设计题目

英文名称：Skyline
中文名称：城市天际线

## 1.2 设计内容

本项目来自 onlinejudge.org （即 Valladolid Programming Contest Site 的OJ题库）

### 英文原文

Draw the skyline of a city according to the data given by the input file.

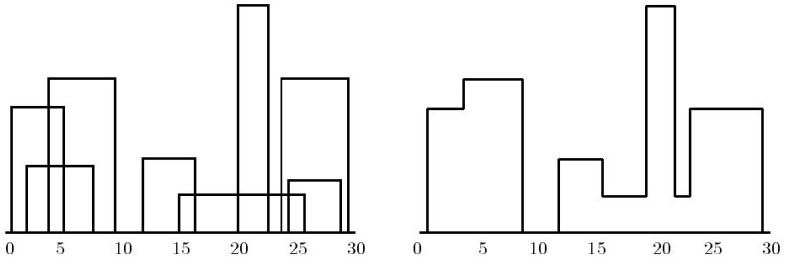
With the advent of high speed graphics workstations, CAD (computer-aided design) and other areas (CAM, VLSI design) have made increasingly effective use of computers. One of the problems with drawing images is the elimination of hidden lines — lines obscured by other parts of a drawing.

You are to design a program to assist an architect in drawing the skyline of a city given the locations of the buildings in the city. To make the problem tractable, all buildings are rectangular in shape and they share a common bottom (the city they are built in is very flat). The city is also viewed as two-dimensional. A building is specified by an ordered triple ($L_i$, $H_i$, $R_i$) where $L_i$ and $R_i$ are the left and right coordinates, respectively, of building $i$ ($0 < L_i < R_i$) and $H_i$ is the height of the building. In the diagram below buildings are shown on the left with triples.

(1,11,5),(2,6,7),(3,13,9),(12,7,16),(14,3,25),(19,18,22),(23,13,29),(24,4,28)

The skyline, shown on the right, is represented by the sequence:

(1,11,3,13,9,0,12,7,16,3,19,18,22,3,23,13,29,0)




**Input**

The input is a sequence of building triples. All coordinates of buildings are integers less than 10,000 and there will be at least one and at most 5,000 buildings in the input file. Each building triple is on a line by itself in the input file. All integers in a triple are separated by one or more spaces. The triples will be sorted by Li, the left x-coordinate of the building, so the building with the smallest left x-coordinate is first in the input file.

**Output**

The output should consist of the vector that describes the skyline as shown in the example above.

In the skyline vector ($v_1$,$v_2$,$v_3$,...,$v_n-2$,$v_n-1$,$v_n$), the $v_i$ such that $i$ is an even number represent a horizontal line (height). The $v_i$ such that $i$ is an odd number represent a vertical line (x-coordinate). The skyline vector should represent the "path" taken, for example, by a bug starting at the minimum x-coordinate and traveling horizontally and vertically over all the lines that define the skyline. Thus the last entry in all skyline vectors will be a '0'.

**Sample Input**

1 11 5
2 6 7
3 13 9
12 7 16
14 3 25
19 18 22
23 13 29
24 4 28

**Sample Output**

1 11 3 13 9 0 12 7 16 3 19 18 22 3 23 13 29 0

**中文翻译**

根据输入文件中给出的数据描绘一个城市的天际线。

随着高速计算机的流行，CAD（计算机辅助设计软件）和其他的领域（CAM，VLSI设计等）已经越发高效地使用着计算机。图形绘制中，怎样消去隐藏的直线——即那些被绘制图形的其他部分遮挡的直线——是一个问题。

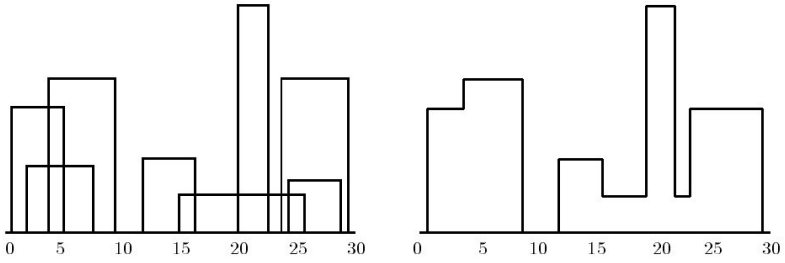
你将要设计一个程序以辅助建筑师通过输入城市中建筑的位置描绘城市的天际线。为了简化这个问题，我们假设所有的建筑都是矩形并且它们

的底部高度相同（这个城市的地面非常的平坦）。同样地，城市也被看作一个二维的平面（正垂面）。在题目中，每座建筑都用一组三元数据表示 $(L_i, H_i, R_i)$ 。其中，$L_i$和$R_i$分别表示建筑 $i$ 两侧边的所在位置的横坐标。下方示意图中的建筑可以用如下的一行三元数据表示：

(1,11,5),(2,6,7),(3,13,9),(12,7,16),(14,3,25),(19,18,22),(23,13,29),(24,4,28)

而图中右半部分展示的天际线则可以用如下的一组数列表示：

(1,11,3,13,9,0,12,7,16,3,19,18,22,3,23,13,29,0)




**输入**

输入的是是一个表示建筑的三元数据的序列。所有的坐标值都是小于10000的整数并且输入文件中建筑的数量不会超过5000座。每一座建筑的坐标数据和它在输入文件中独立成行。同一组数据的两个整数之间用一个或者几个空格分隔。数据已经根据$L_i$从小到大排序，所以有着最小的横坐标的建筑将会第一个出现在输入文件中。

**输出**

输出应该包含如案例中的用来描绘城市天际线的向量。

在天际线向量中 $(v_1,v_2,v_3,...,v_n-2,v_n-1,v_n)$，当$i$是一个偶数时，$v_i$表示天际线的高度而当$i$是奇数时，$v_i$表示天际线中竖线的横坐标。天际线向量实际上表示的是天际线的“路径”。假设一个爬虫根据这些向量的终点水平或者竖直的爬行，那么这个爬虫经过的轨迹就是所需的天际线。因此，向量的最后一个坐标的值为‘0’。

**输入样例**

1 11 5

2 6 7

3 13 9

12 7 16

14 3 25

19 18 22

23 13 29

24 4 28

**输出样例**

1 11 3 13 9 0 12 7 16 3 19 18 22 3 23 13 29 0

### 1.3 开发工具

本项目的开发环境如下：

OS: Ubuntu 22.04.3 LTS on Windows 10 x86_64

Kernel: 5.15.133.1-microsoft-standard-WSL2

CC: gcc (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0

Text Editor: VIM - Vi IMproved 8.2

Debuger: GNU gdb (Ubuntu 12.1-0ubuntu1~22.04) 12.1

### 1.4 应用平台

本项目基于Linux x86_64 ，但是经测试，在 Windows 11 x86_64 上使用编译通过并且可以正常运行。本项目未在 WindowsXP 等操作系统上进行测试，但是考虑到最新版本的 Code::Blocks 自带 GCC 8.1 编译器并且提供了对于 WindowsXP 的支持，而项目中使用的以变量作为数组长度初始化数组等行为最早出现在 C++ 11 标准并且 GCC 从 GCC 4.8.1 开始支持 C++ 11 ，我们可以推测，本项目可以在 WindowsXP 上编译并运行。但是，我们不能确定本项目是否可以支持 Windows 2000 ，Windows 98 等较早的 Windows 版本。

## 2 详细设计

本程序的基本流程如下：

1、通过C++的文件流相关函数，获取输入文件。

2、计算输入文件的行数（考虑到在理想无误的输入文件中，每一行都只有一组输入数据，我们可以根据行数计算输入数据的长度）。

3、建立一个数组，将输入文件中的数据存放在数组当中。

4、逐个分析数组中的数据，并且根据这些数据得到各个坐标处，城市天际线的高度。

5、分析天际线，得到并输出所需要的关键向量。

### 2.1 程序结构

本项目源代码共有三个文件，分别为 file.h, algorithm.h 和 main.cpp ，但是我们可以将三个文件的代码合并到一个文件当中，并不会冲突影响。

分居多个文件的目的是为了使代码结构更加清晰，避免过度复杂，同时也方便将程序模块化，也为之后在实际的应用中方便移植改造。

## 2.2 主要功能

本项目的主要功能如下：

从一个输入文件读取城市建筑的信息，获取其位置和高度。获得城市的天际线数据，并且将数据简化，得到所需要的向量序列，并且在屏幕上输出。本程序具有模块化，可移植等特性，每个函数都可以被单独引用，相互之间依赖少，结构简单，清晰。

程序可以使用g++编译器编译运行。

## 2.3 函数实现

**int countline** 以文件名或文件路径（C语言风格字符串）作为参数。新建一个输入文件流，命名为ReadFile，一个整数表示文件的行数，根据文件路径打开文件。首先使用.fail函数测试文件路径的有效性，若无效，则返回-1。然后，根据文件流每被读取一次，指针就会向后方移动的特点，使用getline函数操作ReadFile，当getline函数返回非0（非 EOF ）时，我们就让n自增一次，代表读取一行。当读取到EOF之后，关闭文件防止内存泄漏，返回n。

**int readfile** 本函数的参数有：C++风格的字符串filename表示文件名，整数型数组d[][3]用于存储数据，整数linenum表示文件行数。在这个函数中，通过数组作为函数形式参数时传递指针常量而非数组数据的特点，实现了返回多值的功能。首先，因为数组的操作不够便利，同时可移植性也不够强，在编写程序时，我先新建三个int ** 指针，名称分别为p，n，x。然后我使用malloc命令给n指针分配内存并将p，x赋值为n以保护n不在后续操作中丢失，造成内存泄露。然后，我通过p将d[n]赋值给*(n+i)。再使用readfile表示文件输入流，整数变量m表示城市的长度（横坐标最大值），利用fin.getline函数，将每行文件：1、输入字符串buff 2、用sscanf输入数据到*(x),*(x+1),*(x+2)，然后令x自增，获取所有数据。在获取数据时，将右侧坐标与m比较，若坐标值大于m，则将m的值变为新的右侧坐标的值。最后，返回m作为城市的长度，也就是天际线的最后一个横坐标 $(v_{n-1})$ 。

**void dataprocess** 本函数需要两个输入参数：int *bitmap 和 int *rectan ，其中，bitmap指针后方有长度大小（单位为整数的存储空间大小）为整个城市长度大小（最大坐标大小）的内存，用于存储在某座标处的最

大高度*rectan指针则代表了一个长度为3的数组，存放给出的三元数据，可以表示出建筑的矩形轮廓。这个函数的功能是描绘每个矩形的天际线，并且消去已经过时的天际线。函数在实现时，将bitmap移到*rectan的位置，然后逐个比较*bitmap和*(rectan+1)的大小，若测得*(rectan+1)更大，则*(bitmap)赋值为*(rectan+1)，并且对*(bitmap)自增，重复直到bitmap等于输入时的bitmap+*(rectan+2)。注：指针加上一个整数表示其后方的第n个存储空间。

**void key_vector** 这个函数的输入值有：int* p 和 int length，这两个参数分别表示城市的天际线和城市的长度。函数的实现原理为：将p[i]和p[i+1]比较，若更改，则输出一个关键向量，最后换行。

**int main** 主函数是整个程序的主体，负责整合，调用几个函数，并且实现相应的功能。输入文件文件名默认为inputfile，但是也可以通过简单的修改，实现输入文件路径的功能。

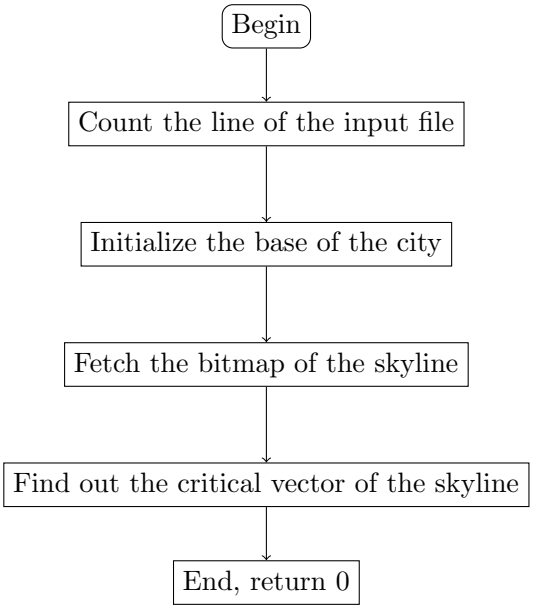
首先，主函数初始化整数linenum，并调用readfile给linenum赋值。

当文件读取正常时，初始化length整数，data[linenum][3]，调用readfile函数给length赋值，并将数据存入data数组。

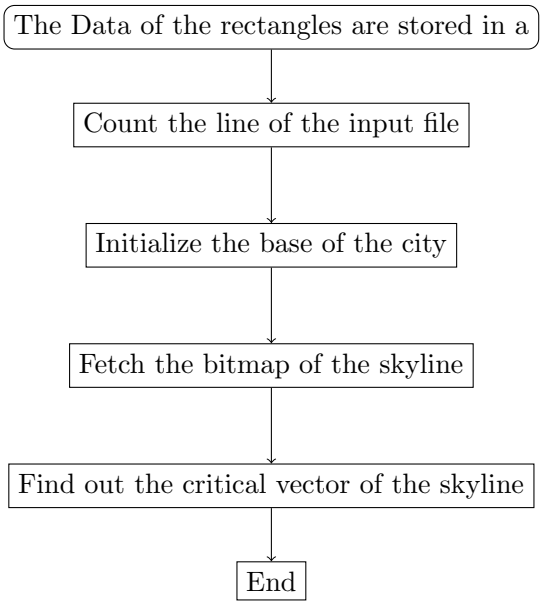
然后我们新建bitmap数组，调用data_process函数将bitmap绘制完成。

最后，调用key_vector函数，输出结果，并返回0表示正常退出。

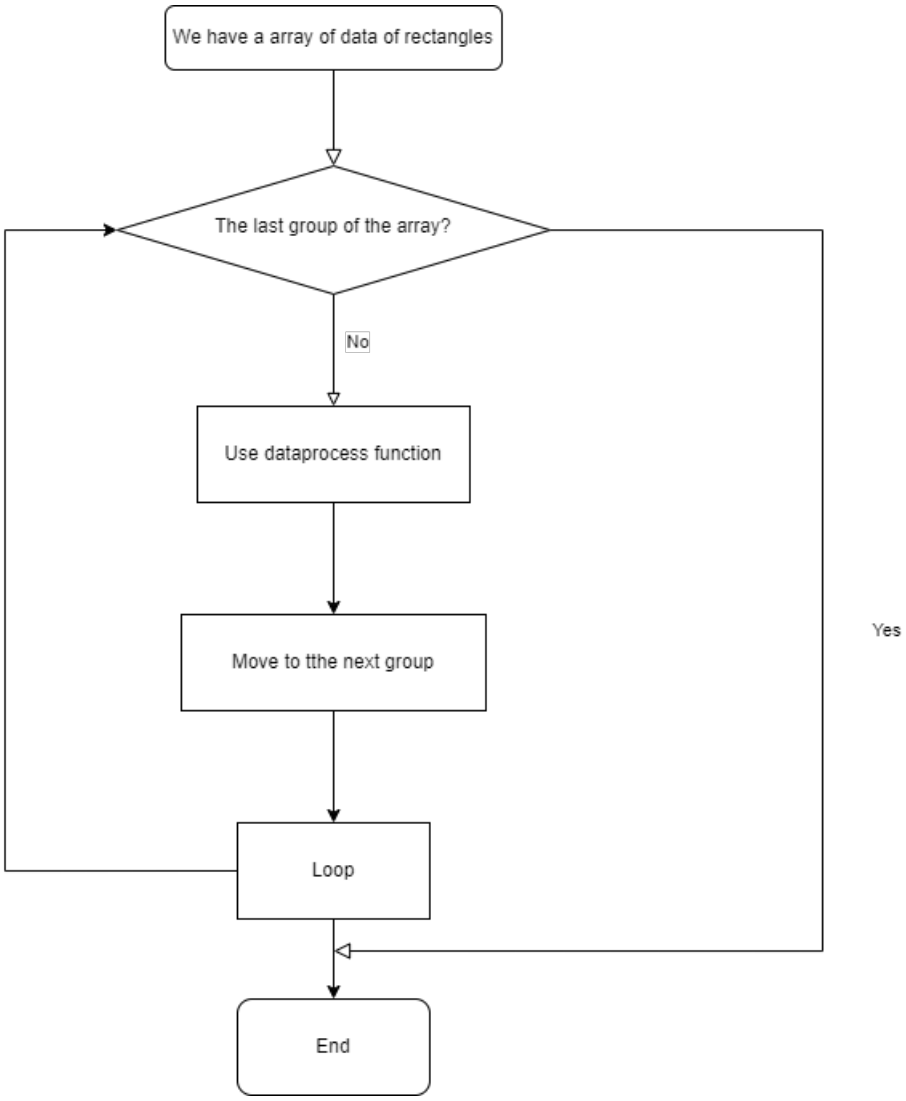
函数的主要流程图示如下：



在“Fetch the bitmap of the skyline”一步中，程序的主要操作较为复杂，涉及循环操作，故图示如下：




循环操作则出现在”Fetch the bitmap of the skyline”中。其流程如下：




**read_data**　这个函数的功能是逐行输出得到的数据，并未使用，仅用于调试。

## 2.4　开发日志

项目的开发过程分为如下几个阶段：

**项目确定**　项目来源于UVA的ACM模拟试题，并且有关问题确实在生产环境中有对应的需求。最终确定本项目遵循模块化原则，将多个函数分离，以便后续对程序代码进行改造和复用，具有可移植性。本项目从一定程度上运用了面向对象的思路，但是受到水平的局限性，故仍然以面向过程为主，

没有使用C++特有的“类”，“多态”，“封装”与“继承”的思想。函数操作也以指针为主，并没有完全符合工程的需求，还有后续改造的空间。

**程序编写** 本次项目的程序代码主要在2023年11月18日完成编写。在编写函数时，为了降低代码复杂度，所有的数组长度都已经在函数参数中被声明，而非调用stdarg标准库。事实上这样的编写方法可以直接地降低代码的复杂度，同时提升运行效率。

**string和char[]** 在开发过程中，由于对于C++风格字符串和C语言风格字符串的用法不熟悉，在程序编写的过程中出现了一些问题。例如在文件流的模板中，在打开文件时文件名只接受字符数组，不接受C++的std::string。因此，后续改造文件时，也需要注意这一点，文件名应该是一个长度足够的字符数组，或者应该是一个分配了足够多内存的字符指针。

**程序的优化** 在主函数中，为了增加程序的稳定性和异常情况应对能力，当输入文件无效时，程序返回-1代表程序异常退出，增加了可靠性。

# 3 程序调试及运行

## 3.1 程序运行结果

程序运行结果如下图：


当输入文件有效时，程序正常执行，否则返回255，即返回-1

加入少量调试代码后，可以清晰地展示出程序的执行流程，效果如下：


## 3.2 程序使用说明

将矩形建筑的数据输入inputfile文件中（注意不是inputfile.txt!）。文件的格式如下：

[$Right\_edge\_1$] [$Height\_1$] [$Left\_edge\_1$]

[$Right\_edge\_2$] [$Height\_2$] [$Left\_edge\_2$]

...

然后输出会被显示在屏幕上，也可以简单修改源代码将其输入到文件当中。

## 3.3 程序开发总结

本次程序设计当中，我较为深入地理解了指针的用法，理解了数组和指针的关系，也明晰了C++中“流”的概念。通过这些知识，我能够更好地掌握C++编程的方法，从而更好地运用在日后的程序设计当中。

数组和指针本是同源，我们可以用指针的方法访问数组，也可以用数组的方式访问指针，二者可以根据需求自由切换。

# 4 附件（源程序）

Listing 1: **file.h**

```
#include<iostream>
#include<fstream>
#include<string>
using namespace std;



//Count the number of lines of a file
int countline(char *filename)
{
    ifstream ReadFile;
    int n=0;
    string tmp;
    ReadFile.open(filename,ios::in);//ios::in 表示以只读的方式读取文件
    if(ReadFile.fail())文件打开失败返回//:0
    {
        return -1;
    }
    else文件存在//
    {
        while(getline(ReadFile,tmp,'\n'))
        {
            n++;
        }
        ReadFile.close();
        return n;
    }
}



//This function can return the max x coordinate value
int readfile(string filename,int d[][3],int linenum){
```

```
34      ifstream fin;
35      fin.open(filename,ios::in);
36      int **n,**p,**x;
37      n=(int **)malloc(linenum*sizeof(int *));
38      p=n;
39      x=n;
40      for(int i=0;i<linenum;i++){
41          *p=d[i];
42          p++;
43      }
44      int m=0;
45      char buff[32];
46      while(fin.getline(buff,sizeof(buff))){
47          sscanf(buff,"%d %d %d",*x,(*x+1),(*x+2)
                );
48          //cout<<buff<<endl;
49          //printf("%d %d %d\n",**n,*(*n+1),*(*n
                +2))测试用代
                码;//
50          if(**x>=m) m=**x;
51          if(*(*x+2)>=m) m=*(*x+2);
52          x++;
53      }
54      free(n);
55      fin.close();
56      return m;
57  }
```

Listing 2: **algorithm.h**

```
1   #include<iostream>
2   using namespace std;
3
4   void data_process(int *bitmap,int *rectan){
5       bitmap+=*rectan;
6       for(int i=*rectan;i<*(rectan+2);i++){
7           if(*bitmap<*(rectan+1)) *bitmap=*(
```

```
            rectan+1);
8           //cout<<*bitmap<<" "<<i<<endl测试用代
                码;//
9           bitmap++;
10      }
11  }测试用代码
12
13  //
14  void read_data(int *p,int n){
15      for(int i=0;i<n;i++){
16          printf("%d.%d ",i,*p);
17          p++;
18      }
19      printf("\n");//
20  }
21
22  void key_vector(int *p,int length){
23      int *a=p;
24      for(int i=1;i<=length;i++){
25          //cout<<a<<endl测试用代码;//
26          if(*a==*(a+1)){
27              a++;
28              continue;
29              //cout<<"*a="<<*a<<" *(a+1)="<<*(a
                    +1)<<endl测试用代
                    码;//
30          }else{
31              printf("%d %d ",i,*(a+1));
32          }
33          a++;
34      }
35      printf("\n");
36  }
```

Listing 3: **main.cpp**

```
1   #include<iostream>
```

```
2  #include"file.h"
3  #include"algorithm.h"
4  using namespace std;
5
6  int main(){
7      int linenum=countline("inputfile");
8      if(linenum!=-1){
9          int data[linenum][3];
10         int length=readfile("inputfile",data,
               linenum);
11         //cout<<linenum<<endl测试用代码;//
12         int bitmap[length+1]={0};
13         //cout<<length<<endl测试用代码;//
14         for(int i=0;i<linenum;i++) data_process
               (bitmap,data[i]);
15         //read_data(bitmap,length+1)测试用代码;//
16         //for(int i=0;i<length;i++) cout<<
               bitmap[i]<<" ";
17         //printf("\n");
18         key_vector(bitmap,length);
19         return 0;
20     }else{
21         return -1;
22     }
23 }
```