Фамилия_____Группа_____28.10.2023

Контрольная работа № 1. Базовые элементы языка C++.

Вариант 1

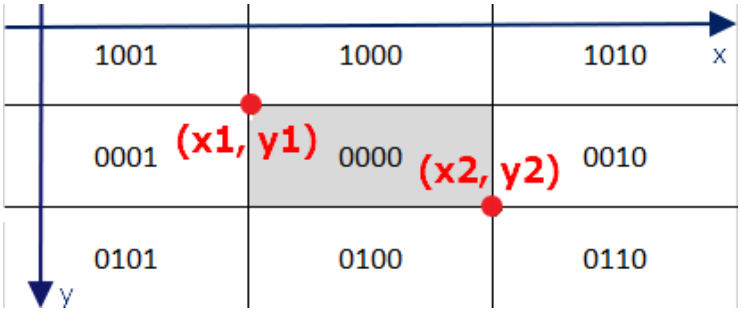
1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*Итого*

Указания:

1. *Решение каждой из задач следует оформить в виде отдельного файла с именем вида* ***kr1_gr1b_v1_ivanov_ivan_z3.cpp. (т.о. Вы будете сдавать 3 файла).***
2. *При реализации используйте контейнеры STL. Не используйте структуры/классы.*
3. *Многомодульный проект: fileIOData.h (*.cpp) visibilityWindow.h (*.cpp), main.cpp*

Задание: разработать приложение для решения задачи об отрезках и окне видимости.

При визуализации объектов на экране компьютера известны размеры экрана, а также размер активного окна. Начало координат – точка (0,0) – верхний левый угол экрана, ось Ox идет слева направо, ось Oy идет сверху вниз. Для анализа того, попадает ли объект в окно видимости используется кодирование частей плоскости с помощью четырех бит. Например, все, что левее окна видимости (т.е. имеет меньшее значение x) кодируется 1 в младшем бите, все что правее (т.е. имеет большее значение x) – 1 во втором бите, все что выше (т.е. имеет большее значение y) – 1 в третьем бите, все что ниже (т.е. имеет меньшее значение y) – 1 в четвертом бите. На рисунке представлены битовые маски для каждой части плоскости.




```
screen.txt
1920 1080
400 500
1500 850
```

```
segments.txt
1400 900 1700 600
0 0 600 350
400 600 1600 200
450 450 350 450
0 0 500 500
1000 800 1000 400
400 500 1500 800
200 900 1700 1000
```

Исходные данные:

1. текстовый файл (например, *screen.txt*) с размерами экрана (screenX, screenY) и с координатами «окна видимости», заданным диагональю (x1, y1), (x2, y2) и являющимся прямоугольником, стороны которого параллельны осям координат.
2. текстовый файл (например, *segments.txt*) с последовательностью пар точек (x1, y1), (x2, y2), определяющих концы отрезка в декартовой системе координат.

Введем псевдонимы для типов данных:

```
using Point = std::pair<int, int>;
using Segment = std::pair<Point, Point>;
```

Реализовать указанные ниже функции:

1. Функция чтения строк из потока `std::vector<std::string> read(std::istream& theStream) noexcept`.
2. Перегрузка функции п.1. Функция чтения строк из файлового потока `std::vector<std::string> read(const std::string& filePath)`, которая вызывает функцию п.1. В случае отсутствия файла выбрасывать исключение `std::invalid_argument("File name is wrong: " + filePath)`.
3. Функция парсинга строки в числа `std::vector<int> parser(const std::string& str)`; При реализации учитывайте то, что в строке могут быть заданы координаты нескольких точек. Если в строке нечетное количество чисел, т.е. невозможно создать точку, необходимо выбрасывать исключение `std::invalid_argument("It is not enough data")`;
4. Перегрузка функции п.3. Функция парсинга коллекции строк `std::vector<Point> parser(const std::vector<std::string>& container)`; в координаты, которая для каждой парсинга каждой строки вызывает функцию п.3.
5. Функция создания «экрана» на основе набора точек и нормализация параметров окна видимости `std::array<Point, 3> buildScreen(const std::vector<Point>& list)`;

а) Если коллекция содержит не три точки, то выбрасываем исключение std::invalid_argument("It is not enough screen data");

б) Размеры экрана (первая пара чисел) – неотрицательные числа, иначе выбрасывать исключение std::invalid_argument("Window dimensions must be positive").

в) Координаты, задающие диагональ окна видимости должны быть определены так, чтобы это были левая верхняя и нижняя правая точка, как на рисунке (x1,y1) <= (x2, y2). Если это не так, то переопределите координаты диагонали так, чтобы окно видимости осталось прежним, но при этом было выполнено требование, предъявленное к диагонали.

г) Окно видимости должно находиться в пределах экрана, это не так, то нужно «обрезать» границы окна видимости так, чтобы окно видимости было видимым на экране. Если окно видимости целиком находится вне экрана, то выбрасывать исключение std::invalid_argument("Visisbility window is out the screen").

6. Функция определения кода точки (см.рисунок) uint8_t caclCode(const std::array<Point, 3>& screen, const Point& point) noexcept;. *Замечание: учитывайте, что uint8_t определено:typedef unsigned char uint8_t; поэтому, при выводе на экран отображается как символ с соответствующим кодом. Чтобы увидеть числовое значение, нужно будет конвертировать.*
7. Перегрузка функции п.6. Функция определения кода всех точек концов отрезков (для получения коллекции points надо будет получить список концов отрезков на основе функций п.2 и п.4) std::map<Point, uint8_t> caclCode(const std::array<Point, 3>&screen,const std::vector<Point>& points) noexcept;
8. Функция создания отрезков std::vector makeSegments(const std::vector<Point>& points) noexcept; Коллекция points содержит на i-й и i+1 позиции начальную и конечную точки отрезка, для i от 0 до конца коллекциис шагом 2.
9. Функция определения уникальных точек std::set<Point> findUniquePoints(const std::vector<Point>& points) noexcept;
10. Функция определения уникальных отрезков std::set findUniqueSegments(const std::vector& segments) noexcept; Функция выполняет «нормализацию» множества отрезков: делаем отрезки ориентированными, началом отрезка будем считать точку, у которой абсцисса меньше, т.е. x1 < x2, если x1 = x2, то сравниваем y1 и y2. Началом будет точка, у которой y меньше. Вырожденные отрезки исключаем не включаем во множество уникальных отрезков.
11. Функции нахождения списка точек лежащих в окне видимости и вне окна видимости
    std::set<Point> findPointsInWindow(const std::map<Point, uint8_t>& codePoints) noexcept;
    std::set<Point> findPointsOutsideWindow(const std::map<Point, uint8_t>& codePoints) noexcept;
12. Функции определения того, что отрезок находится в окне видимости bool segmentIsInside(const Segment& segment, const std::map<Point, uint8_t>& codePoints) noexcept;
13. Функции определения того, что отрезок находится вне окна видимости bool segmentIsOutside(const Segment& segment, const std::map<Point, uint8_t>& codePoints) noexcept;
14. Функция нахождения коллекции отрезков, находящихся в окне видимости std::set findSegmentsInWindow(const std::set& segments, const std::map<Point, uint8_t>& codePoints) noexcept;
15. Функция нахождения коллекции отрезков, находящихся в окне видимости std::set findSegmentsOutsideWindow(const std::set& segments, const std::map<Point, uint8_t>& codePoints) noexcept;