Junioraufgabe 1: Landnahme

Tobias Nöthlich - Teilnahme-ID: 6396

29.11.2015

Inhaltsverzeichnis

Teil I
Lösungsidee

Ein *Grundstück* ist ein nach den Himmelsrichtungen ausgerichtetes, beliebig großes Rechteck, welches, um akzeptiert zu werden keinerlei Überschneidungen mit anderen Grundstücken haben darf. Es wird durch mehrfache Verwendung des Zeichens "+" graphisch dargestellt. Eine *Überschneidung* liegt vor, wenn sich ein Teil eines Grundstücks G_2 innerhalb der Grenzen eines vorher festgelegten Grundstückes G_1 befindet. dabei spielt es keine Rolle ob G_2 ganz oder nur teilweise von G_1 eingeschlossen wird. Zur Überprüfung auf Überschneidungen wird ein zweidimensionales *Feld* definiert, auf welchem die Grundstücke angelegt werden müssen.

Grundstück

Überschneidung

Feld

Im Wesentlichen müssen nun innerhalb einer Funktion *Berechnung()* die zwischen den vom Benutzer eingegebenen Punkten liegenden X- und Y-Werte bestimmt und gespeichert werden. Die Funktion kann später, sollte das Grundstück genehmigt werden, die X- bzw. Y-Werte nutzen um die Punkte die es umfasst festzulegen. Berechnung()

Teil II
Umsetzung

1 Variablen

Zur Umsetzung der Lösungsstrategie habe ich folgende Variablen global definiert:

```
1  int Anzahl;                  //Anzahl der zu testenden Grundstücke
2  int xCoord1, yCoord1, xCoord2, yCoord2; //Koordinaten der Punkte
3  int breite, hoehe;           //Breite und Höhe der Grundstücke
4  std::vector<int> X, Y;       //Alle X- / Y-Koordinaten eines Rechtecks
5  char Feld[20][20] = { 0 };   //Das Feld
6  void Berechnung();
7  void Ausgabe();
8  bool erlaubt = true;
```

2 int main()

Zur Lösung dieses Problems habe ich ein zweidimensionales Array namens *Feld[20][20]* angelegt, welches 20 Reihen und Spalten enthält. Ich habe es nicht größer gewählt, da man es in dieser größe bequem ausgeben kann und die Aufgabe einen Maximalwert von 6 auf der X-Achse verlangt. In der Funktion *int _tmain(int argc, _TCHAR* argv[]){...}* ist die Eingabe der beiden benötigten Punkte untergebracht. Außerdem wird durch sie die Funktion *Berechnung()* aufgerufen und das Ergebnis, ob ein Claim akzeptiert wurde oder nicht ausgegeben. Durch den Aufruf der Funktion *Ausgabe()* wird desweiteren der Inhalt von *Feld* auf dem Bildschirm ausgegeben. *X.clear()* und *Y.clear()* leeren die Vektoren um sie für das nächste Grundstück neu befüllen zu können.

3 void Berechnung()

In der Funktion *void Berechnung(){...}* werden aus den eingegeben Werten die X-Werte welche zwischen X_1 und X_2 liegen erzeugt und in den *std::vector<int> X* gespeichert. Gleiches geschieht mit den Y-Werten, mit dem Unterschied, dass diese in einen *std::vector<int>* namens *Y* gespeichert werden. Besagte X- und Y-Werte werden daraufhin benutzt um zu testen ob der momentane Punkt des zu testenden Grundstücks innerhalb der Grenzen eines schon festgelegten Grundstücks liegt. Eine Ausnahme bildet hierbei die äußerste Grenze eines Grundstücks, da diese gleichzeitig als Grenze eines neuen Grundstücks gelten kann. Sollte keine Überschneidung

vorliegen, so wird das neue Grundstück Stück für Stück in das Feld geschrieben, sodass an dieser Stelle kein zweites Grundstück bewilligt werden kann.

4 void Ausgabe()

Die Funktion *void Ausgabe()* wird aufgerufen nachdem feststeht ob ein Grundstück akzeptiert wurde oder nicht. Sie dient dazu das Array Feld auf dem Bildschirm anzuzeigen. Dies wird durch zwei for-Schleifen ermöglicht, welche jeweils die auszugebende Position im Array *Feld* angeben und nach jeder Arrayzeile einen Zeilenumbruch einfügt. Aufgrund der Eigenschaft eines Arrays die Position [0][0] in der oberen linken Ecke zu haben ist die Ausgabe nicht mit einem Koordinatensystem gleichzusetzen.

Teil III
Beispiele

Eingabe	Ausgabe	akzeptiert / abgelehnt
(2/3), (5/5)	(2/3), (5/5)	akzeptiert
(1/2), (4/4)	(1/2), (4/4)	abgelehnt
(3/1), (6/3)	(3/1), (6/3)	akzeptiert

Tabelle 1: Aufgabenbeispiel

Für das in der Aufgabe vorgegebene Beispiel (vgl. Tabelle 1) generiert mein Programm die folgende Ausgabe:

Eingabe	Ausgabe	erwartetes Ergebnis
(1/2),(3/4)	(1/2),(3/4)	akzeptiert
(2/5),(4/8)	(2/5),(4/8)	akzeptiert
(9/12),(15/17)	(9/12),(15/17)	akzeptiert

Tabelle 2: selbst erdachtes Beispiel

Für ein zweites selbst erdachtes Beispiel habe ich die Werte aus Tabelle 2 genommen. Die Ausgabe war gleich dem erwarteten Ergebnis.

Teil IV
Quellcode

Bwinf_Landnahme.cpp

```
1  // Bwinf_Landnahme.cpp : Definiert den Einstiegspunkt für die Konsolenanwendung.
2  //
3
4  #include "stdafx.h" // vorkompilierter Header
5  #include <iostream> //Ein-/Ausgabe
6  #include <vector>  //Vektorbefehle
7  #include <cstdlib> //System("...")-Befehle
8
9
10
11 int Anzahl;                        //Anzahl der zu testenden Grundstücke
12 int xCoord1, yCoord1, xCoord2, yCoord2;      //Koordinaten der Punkte
13 int breite, hoehe;                  //Breite und Höhe der Grundstücke
14 std::vector<int> X, Y;               //Alle X- / Y-Koordinaten eines Rechtecks
15 char Feld[100][100] = { 0 };            //Das Feld
16 void Berechnung();
17 void Ausgabe();
18 bool erlaubt = true;
19
20
21 int _tmain(int argc, _TCHAR* argv[])
22 {
23   std::cout << "Wie viele Grundstuecke moechten Sie anlegen?\n";
24
25   std::cin >> Anzahl;
26
27   system("cls");
28
29   for (Anzahl > 0; Anzahl--;)
30   {
31
32     std::cout << "----------2\n"
33       << "|----------|\n"
34       << "|----------|\n"
35       << "1----------\n\n";
36     std::cout << "Bitte geben Sie nacheinander die (ganzzahligen) Koordinaten der Punkte 1/2 ein.
         (x,y <= 100) \n";
37
38     std::cin >> xCoord1
39       >> yCoord1
40       >> xCoord2
41       >> yCoord2;
42
43
44     Berechnung();
45
46     std::cout <<"Punkt 1: (" << xCoord1 << "/"
47       << yCoord1 << ")\nPunkt 2: ("
48       << xCoord2 << "/"
49       << yCoord2 << ")\n"
50       <<"Breite: " << breite <<"\n Hoehe: "
51       << hoehe << "\n";
52       std::cout<< "\n";
53
54       if (erlaubt == true)
55     {
56       std::cout << "Claim akzeptiert!\n";
57     }
58
59     if (erlaubt == false )
```

```
60        {
61          std::cout << "Claim abgelehnt!\n";
62        }
63
64        Ausgabe();
65
66        X.clear();
67        Y.clear();
68
69        system("pause");
70      }
71      return 0;
72    }
73
74    void Berechnung()
75    {
76      /*VARIABLEN*/
77      erlaubt = true;
78      breite = xCoord2 - xCoord1; //Breite des Rechtecks berechnen
79      hoehe = yCoord2 - yCoord1; //Höhe des Rechtecks berechnen
80      int breitenCheck = xCoord1;
81      int hoehenCheck = yCoord1;
82
83
84      /*X UND Y VECTOR BERECHNEN*/
85
86      for (xCoord1; breitenCheck <= xCoord2; breitenCheck++)
87      {
88        X.push_back(breitenCheck);
89      }
90
91      for (yCoord1; hoehenCheck <= yCoord2; hoehenCheck++)
92      {
93        Y.push_back(hoehenCheck);
94      }
95
96
97      /*ÜBERPRÜFUNG OB ÜBERLAPPUNG VORLIEGT*/
98
99      for (int i = 0; i < X.size(); i++)
100     {
101       int x = X[i];
102
103       for (int i = 0; i < Y.size(); i++)
104       {
105         int y = Y[i];
106
107         if (Feld[y][x] == '+' && Feld[y - 1][x] == '+' && Feld[y][x - 1] == '+' && Feld[y + 1][x]
                == '+' && Feld[y][x+1] == '+') //damit die Grenzen nicht als Sperre gelten
108         {
109           erlaubt = false;
110         }
111
112
113       }
114     }
115     /*SPEICHERN DER BELEGTEN PUNKTE*/
116     if (erlaubt==true)
117     {
118         for (int j = xCoord1; j <= xCoord2; j++)
119         {
120           for (int i = yCoord1; i <= yCoord2; i++)
121           {
122             Feld[i][j] = '+';
123           }
124         }
125       }
126     }
```

```
127
128
129 void Ausgabe()
130 {
131
132   //system("cls");
133
134   for (unsigned int i(0); i != 20; ++i)
135   {
136     for (unsigned int j(0); j != 20; ++j)
137     {
138       std::cout << Feld[i][j] << " "; // Elemente einer "Arrayzeile" nebeneinander ausgeben
139     }
140     std::cout << "\n"; // Jede neue "Arrayzeile" in neuer Zeile ausgeben
141   }
142 }
```