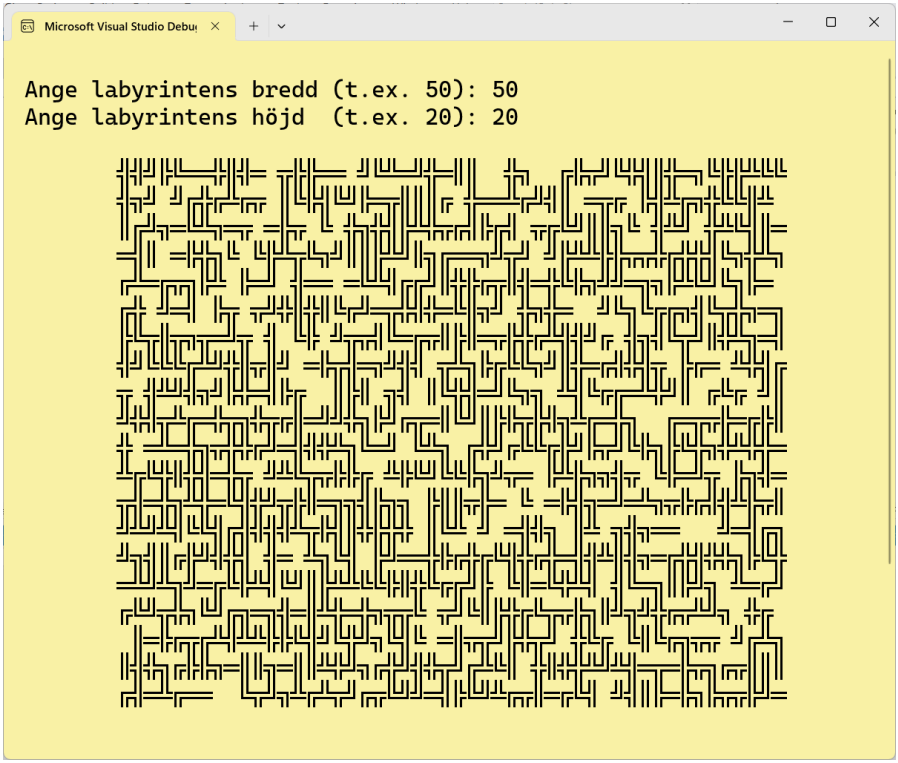
Inlämningsuppgift 2

Labyrinten Visst är det roligt att med våra C++ kunskaper hittills kunna skriva ett program som ritar en labyrintartad figur i konsolen som kan se ut så här:



Detta är förstås inte någon riktig labyrint. För en sådan skulle kräva mycket mer. En riktig labyrint med in- och utgång osv. skulle kunna ingå i ett spelprojekt med grafiska finesser, färger osv. En sådan avancerad figur kan inte ritas i konsolen.

Vår uppgift går snarare ut på att i *text mode* ”rita” en *labyrintartad figur* som är slumpmässigt ihopsatt av mellanslaget och ett antal tecken som vi kallar för *dubbla linjegrafiktecken (LGT)*. I figuren ovan är dessa tecken slumpade i en 2D utskrift, en slags tabell eller matris med 50 rader och 20 kolumner. I koden åstadkommer man detta med en nästlad **`for`**-loop, där den yttre loopen skriver ut raderna och den inre loopen kolumnerna, se avsn. ***6.8 Nästlade for-satser***, sid 157. Alla tecken i figuren ovan är slumpvist valda bland de dubbla LGT och mellanslaget. Därför borde varje körning av programmet generera en lite annorlunda labyrintartad figur.

Du kan gärna försöka med en egen algoritm att skriva ett C++ program som ritar en sådan figur. Men i instruktionerna som följer nedan har du i alla fall ett förslag till en enkel algoritm (***Steg 1-5***) som fungerar.

Algoritmen

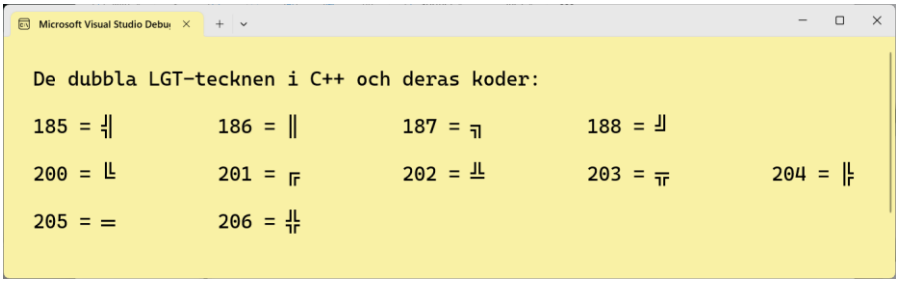
Steg 1 Bekanta dig med hantering av tecken i C++ inkl. explicit typkonvertering (sid 120), genom att experimentera med programmet **`Int2char`** som behandlades på lektion 11 (sid 121):

```
// Int2char.cpp
// Ger tecknet till en inmatad kod
// Representation av tecken med heltalskoder
#include <iostream>
using namespace std;

int main()
{
   int code;
   cout << "\nMata in ett heltal: ";
   cin >> code;
   cout << "\nDet inmatade heltalet är " << code <<
        " och är koden till tecknet " << (char) code << "\n";
}
```

Experimentera med **`Int2char`** genom att mata in koderna **`185-188`** och **`200-206`**.

Steg 2 För få en översikt över alla elva dubbla LGT samt deras koder i C++ skriv ett program som producerar följande utskrift:



Dessa tecken finns i den utvidgade delen av teckentabellen (utöver standard ASCII) och används för att rita raka linjer, ramar, tabeller, skisser osv i en textbaserad miljö *(text mode)*. De kan användas tillsammans med mellanslaget för att rita labyrinten. Jämför koderna även med utskriften till programmet **`AsciiFor`** (sid 155).

Steg 3 Repetera hantering av slumptal i avsn. ***4.9 Hantering av slumptal*** (sid 95), speciellt om ***Slumptal inom ett intervall*** (sid 96).

Steg 4 Skriv ett program som med hjälp av C++:s slumpgenerator och en nästlad **`for`**-sats ritar labyrinten, en slumpmässigt ihopsatt figur bestående av de dubbla LGT samt mellanslaget.

Steg 4 i detalj

Här följer i själva verket hur ***Steg 4*** kan realiseras:

Deklarera en teckenvariabel **`letter`** och en heltalsvariabel **`randNo`**. Initiera **`letter`** till mellanslaget, dvs **`' '`**. Låt **`randNo`** anta slumpvärden mellan **`0`** och **`11`** med satsen:

```
randNo = rand() % 12;
```

Fortsätt med följande **`if`**-satser:

Om **`randNo`** blir **`0`**	ska **`letter`** tilldelas	1:a LGTs teckenkod.
Om **`randNo`** blir **`1`**	ska **`letter`** tilldelas	2:a LGTs teckenkod.
Om **`randNo`** blir **`2`**	ska **`letter`** tilldelas	3:e LGTs teckenkod.
.	.	
.	.	
.	.	
Om **`randNo`** blir **`9`**	ska **`letter`** tilldelas	10:e LGTs teckenkod.
Om **`randNo`** blir **`10`**	ska **`letter`** tilldelas	11:e LGTs teckenkod.
Om **`randNo`** blir **`11`**	ska **`letter`** tilldelas	mellanslaget, dvs **`' '`**.

Numreringen av LGT-teckenkoderna avser den ordning som är föregiven i teckentabellen, se utskriften på förra sidan. I övrigt fungerar vilken numrering som helst.

Observera att LGT-teckenkoderna är av typ **`int`**, medan variabeln **`letter`** är av typ **`char`**. Vid tilldelningen konverteras **`int`** automatiskt till **`char`**. Vid utskriften med **`cout`** skrivs ut tecknet, inte koden. Mellanslaget (**`' '`**) däremot är från början av typ **`char`**, så att vi inte behöver bry oss om koden. Se upp att **`''`** inte är mellanslaget utan den tomma strängen och ger kompileringsfel.

Skriv ut **`letter`**, så att du får ETT tecken, antingen ett LGT eller mellanslaget. Testa även om du vid varje körning får *olika* LGT eller mellanslaget.

Först när allt detta fungerar låt tilldelningen av variabeln **`randNo`** samt de 12 **`if`**-satserna ovan ingå i en enkel *loop*, t.ex. en **`for`**-sats, för att rita labyrinten.

Ersätt den enkla loopen med en nästlad **`for`**-sats och lägg in ett radbyte mellan den inre och yttre slingan för att kunna styra labyrintens storlek (höjd och bredd) vid varje körning. Låt användaren ange höjd och bredd.

Algoritmen ovan är enkelt genomförbar, men inte den mest eleganta lösningen. Har du klarat av den kan du gärna gå vidare till följande:

Extrauppgift (frivilligt)

Ersätt de 12 **`if`**-satserna ovan med en annan konstruktion: Samla alla dina LGT-koder och mellanslaget i en *array* och låt slumpen ta ut tecken ur denna array. För utskriften kan du fortsätta med att använda nästlad **`for`**-sats. Det blir en kortare och elegantare kod.