18.06.20 19:06

CSCI 125
Introduction to Computer Science and Programming II
Lecture 4: *Village of The Sorcerer*

Jetic Gū

2020 Summer Semester (S2)

Overview

- Focus: Basic C/C++ Syntax
- Architecture: Linux/Unix OS
- Core Ideas:
    1. Compilation
    2. About the task
    3. Other useful knowledge

C++ Compilation

Preprocessor Directives
Your own library

C++ Code

- Preprocessor Directives

  `#include`

- The rest, such as function declarations (including `main()`), variable declarations, function prototypes

Review

Preprocessor Directives[1]

- Source file inclusion (`#include`)
- Macro definitions

  `#define`

  - Predefined macro names

* *Conditional inclusions*

  * `#ifdef, #ifndef, #if, #endif, #else and #elif`

* *Line control (`#line`), Error directive (`#error`)*

* Not required

1. http://www.cplusplus.com/doc/tutorial/preprocessor/

Concept

Source file inclusion

- Source file inclusion (`#include`)
- For system/C++ standard libraries

```
#include <HEADER>
```

- Your own stuff: everything you include here will be compiled together, just like in the same file

```
#include "MyHeader.h"      // macros, function prototypes
#include "MyHeader.hpp"    // less conventional header[1]
#include "MyHeader.cpp"    // actual source code
```

Concept

1. Some people uses .hpp instead of .h to differentiate between C++ headers and C headers. This is very uncommon though.

Source file inclusion

- Source file inclusion (`#include`)
- For system/C++ standard libraries

```
#include <HEADER>
```

- Your own stuff: everything you include here will be compiled together, just like in the same file

```
#include "MyHeader.h"      // macros, function prototypes
#include "MyHeader.hpp"    // less conventional header[1]
#include "MyHeader.cpp"    // actual source code
```

Concept

1. Some people uses .hpp instead of .h to differentiate between C++ headers and C headers. This is very uncommon though.

Macro definitions

- Syntax

```
#define identifier replacement
#define identifier(...) replacement  // with parameters
```

- Compilation
  - Replaces every occurrence of identifier with replacement

Concept

Macro definitions

- Replacement criteria
  Has to be identifier
- Is this going to work?
- What will the output be?
  `int` or `Cheese`?

```
#include <iostream>
#define Cheese int
using namespace std;
Cheese main() {
   cout << "Cheese" << endl;
   return 0;
}
```

Concept

Macro definitions

- Common macro functions

```
#define max(x, y) (x>y?x:y)
#define min(x, y) (x<y?x:y)
```

- `?` operator, e.g. `(x>y?x:y)`
  - `(condition ? returnIfTrue: returnIfFalse)`

Concept

Macro definitions

- Compiler sees identifier, replaces
- Equivalent to
  - `cout << (13>14?13:14);`

```
#include <iostream>
#define max(x,y) (x>y?x:y)
using namespace std;
int main() {
  cout << max(13,14);
  cout << endl;
  return 0;
}
```

Concept

Predefined macro names

- `__FILE__`: name of the source file
- `__DATE__`: date
- `__TIME__`: time
- etc.

```
#include <iostream>

using namespace std;

int main() {

  cout << "Line number " << __LINE__;

  cout << " of file " << __FILE__ << ".\n";

  cout << "Its compilation began " << __DATE__;

  cout << " at " << __TIME__ << ".\n";

 return 0;

}
```

Concept

Compilation Steps

1. Preprocessing
    - Process preprocessing directives
2. Compilation
    - Parse your code, analyse syntax, perform optimisation (for you)
    - Product: object files (.o)
3. Linking
    - Replace all references of variables and functions (such as `scanf` in `cstdio`) with executable binary code
    - Product: executable files

Concept

Compilation

- I’ve never seen object files, is it not needed?
- When you do this

```
g++ sol.cpp -o sol
```

  - This is automatically **compiling** and **linking**

Concept

Compilation

- Let’s say you have your own library
  - `myheader.h`: header for function prototypes to be used in `main.cpp`
  - `myheader.cpp`: implementation for functions declared in `myheader.h`
    - Must `#include "myheader.h"`
  - `main.cpp`: `#include "myheader.h"`, and `main` function uses function from this header

Concept

Compilation

- Compilation
  - `g++ -c myheader.cpp -o myheader.o`
    - Only perform **Preprocessing** and **Compilation** here
  - `g++ main.cpp myheader.o`
    - **Preprocessing** + **Compilation** + **Linking**

Concept

P2
VoTS

Village of The Sorcerer

Function Practice
and a lot of fun

Summary

P2
VoTS

Village of The Sorcerer

- Released practice problems so far
  - P014-P018
- An RPG game
- You will be able to play the full version at the end of this semester
  - And it’s written by YOU!

Concept

P2
VoTS

Village of The Sorcerer

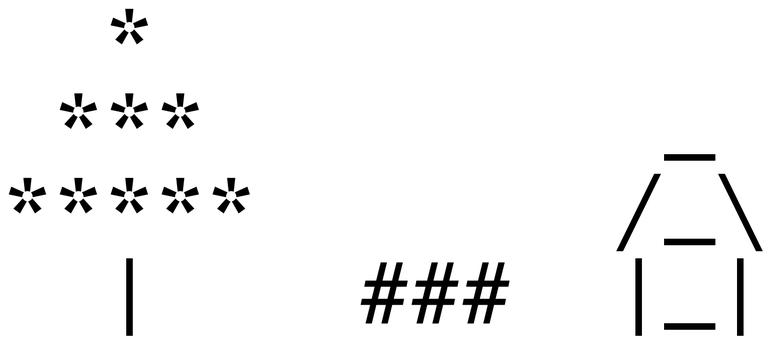
- This is a map
- A character can move inside the map
  - Controlled using keyboard

Concept

P2
VoTS

Village of The Sorcerer

- Objects
  - # this is a fence
  - Trees
  - Houses



Concept

P2
VoTS

Village of The Sorcerer

- Why this exercise?
  - You will need to create a lot of functions to efficiently implement this game
    - I will teach you how
  - You will eventually need to divide the game into modules, and implement each one separately
    - The best way to learn C++ is to write more code!

Concept

P2
VoTS

Village of The Sorcerer

- For P014-P018
  - I will post a list of all the Global Variables and Function Prototypes, as well as their meanings, as well as more in class tutorials
  - What you have to do
    - Study each questions
    - Understand how functions are called and what each one does
    - Submit, and make sure it is working

Concept

Village of The Sorcerer

P014-P018

Summary

P3
Useful stuff

Village of The Sorcerer

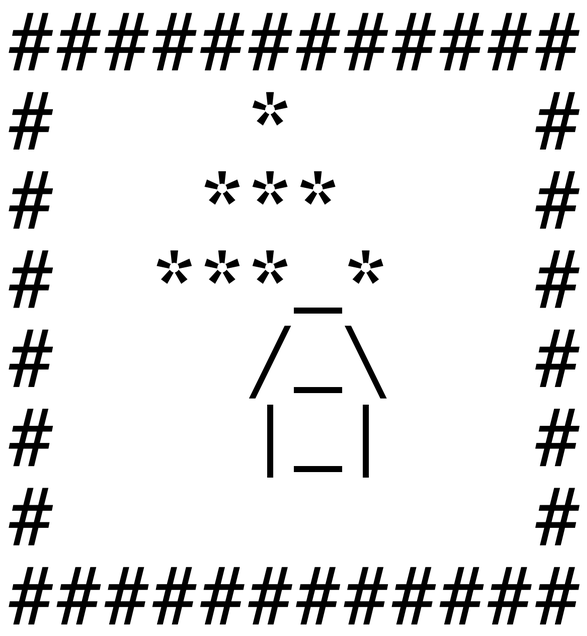
- *P014:* Read in map description, and correctly print a map on screen
- *P015:* Read in multiple map descriptions, and correctly print maps on screen
- *P016:* Character movement
- *P017:* Character movement with restrictions
- *P018:* Character movement across multiple levels

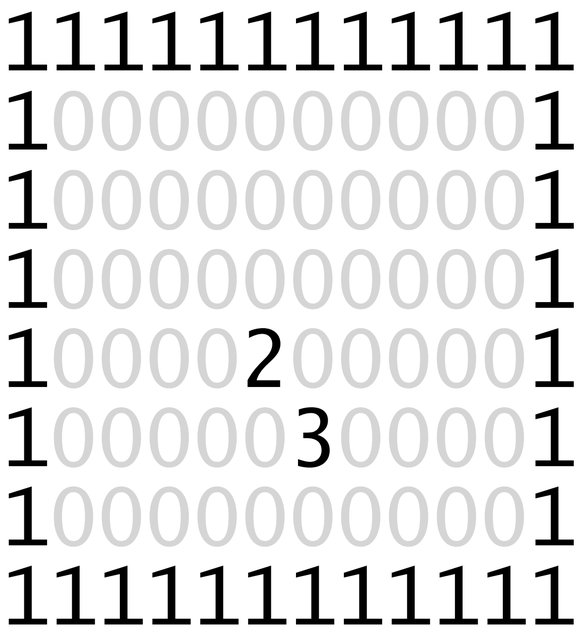
Technical

P3
Useful stuff

VoTS maps

- A map is represented by an N x M matrix
- Each element represents an object
  - This was the way most games store maps in the 90s! including Red Alert, Civilisation, etc.! Now it's just with more info.





Technical

P3
Useful stuff

VoTS

- 1 is for fences, denoted by Hashtags

```
1111111111111
1000000000001
1000000000001
1000000000001
1000000000001
1000020000001
1000003000001
1000000000001
1111111111111
```

Technical

P3
Useful stuff

VoTS

- 2 is for trees
- 1 vertical slash, 9 asterisks

Technical

P3
Useful stuff

VoTS

- 3 is for houses
- 3 underscores, 2 vertical slash, 1 left slash, 1 right slash

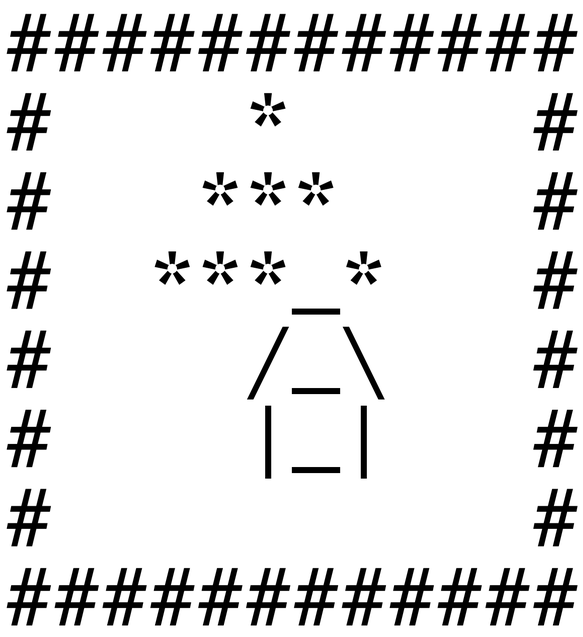
Technical

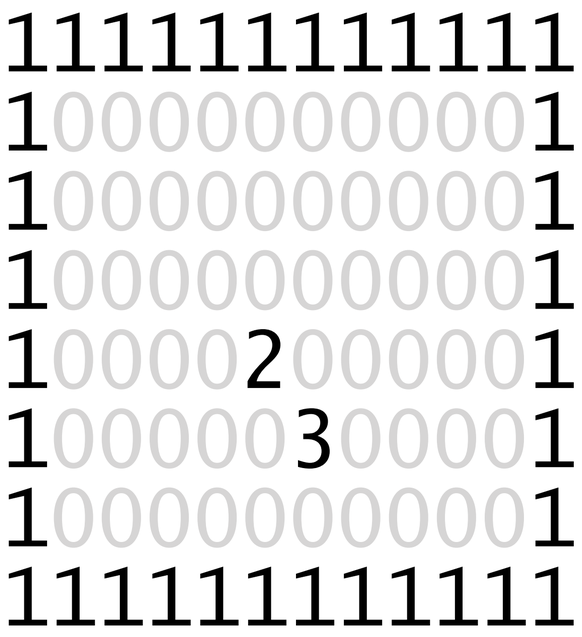
P3
Useful stuff

VoTS maps

- **Cover**, trees and houses **in front of** other objects blocks the view

Greater X coordinate, or if same X greater Y coordinate



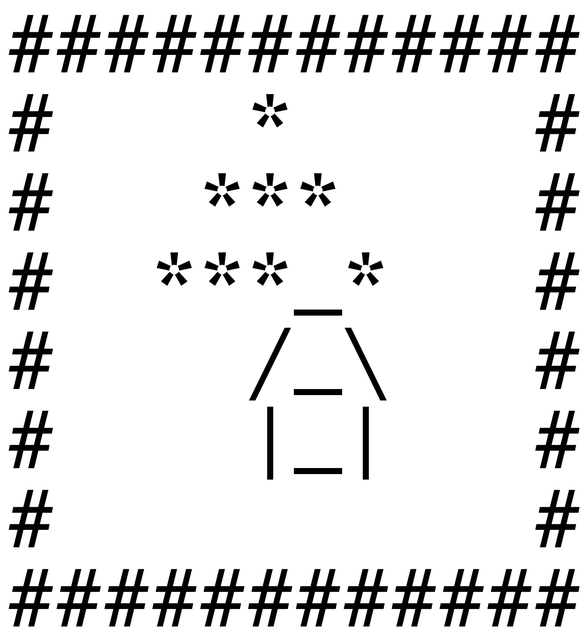


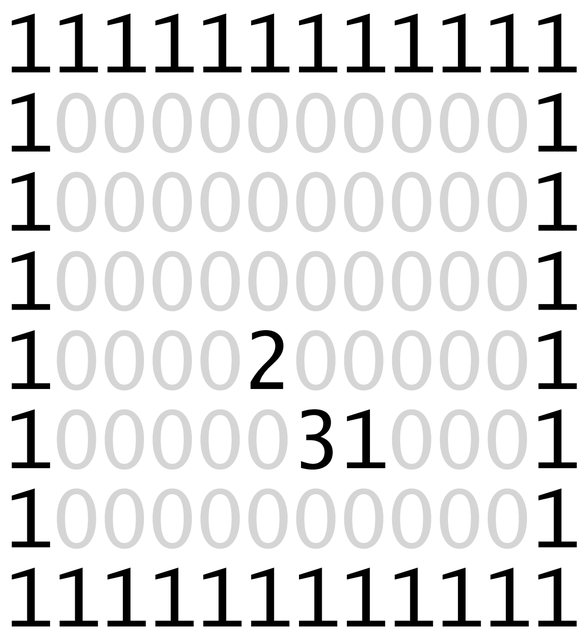
Technical

P3
Useful stuff

VoTS maps

- Fences cannot block trees and houses





Technical

P3
Useful stuff

P014-P015: VoTS 1-2

- Map printing
- Recommended Public variables and functions

```
1. #include <iostream>
2. using namespace std;
3. int n, m;


5. void drawTree(char a[][100], int x, int y);
6. void drawHouse(char a[][100], int x, int y);
7. void printMap(int map[][100]);
```

Technical

P3
Useful stuff

P014-P015: VoTS 1-2

1. `void drawTree(char a[][100], int x, int y);`

- draw a tree to position `x, y` of output bitmap `a`

2. `void drawHouse(char a[][100], int x, int y);`

- draw a house to position `x, y` of output bitmap `a`

3. `void printMap(int map[][100]);`

- Print the map specified by `map`, using the drawing functions
    - Creates bitmap `a`

Technical

P3
Useful stuff

P016-P017: VoTS 3-4

- With character movements and restrictions
- Recommended Public variables and functions

```
#include <iostream>
using namespace std;
int n, m;


void drawTree(char a[][100], int x, int y);
void drawHouse(char a[][100], int x, int y);
void printMap(int map[][100], int x, int y);
int checkMove(int map[][100], int x, int y);
void movePlayer(int map[][100], int& x, int& y, int dir);
```

Technical

P3
Useful stuff

P016-P017: VoTS 3-4

7. `void printMap(int map[][100], int x, int y);`

- Augment print function with player position

8. **`int checkMove(int map[][100], int x, int y);`**

- Check if the player can move to position `x, y` of `map`

- Returns result of the check (true or false)

9. **`void movePlayer(int map[][100], int& x, int& y, int dir);`**

- Moves the player from `x, y` to a direction specified by `dir`

Technical