717310: Game Programming

Steffan Hooper

Wednesday, 22 July 2015

Overview

- Procedural C++
  - An introduction to Programming in C++
- Exercises

Game Programming

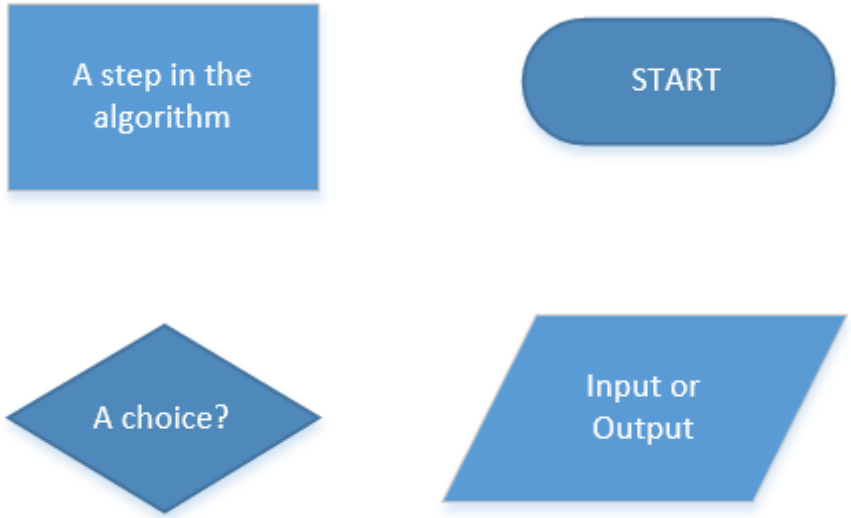
- Game Programming:
  - The practice of video game software development.
    - Simulation, Artificial Intelligence, Computer Graphics, Physics Simulation, Audio, Input, etc…
  - C++ is widely used…
    - Object oriented…
    - Native binary compilation and cross-platform support.
    - Existing middleware and APIs:
      - Game engines and library modules.

History of Computer Games

- Programming Languages Used In Games:
  - Assembly: Optimisation…
  - C++: “AAA” Development, Engines, Consoles, Cross-Platform…
  - C#: XNA, PlayStation Mobile, Tools…
  - Java: Mobile, Android, Tools…
  - Lua: Scripting…
  - Python: Tools…
  - GLSL, HLSL, PSSL, MetalSL: Shader Languages for Computer Graphics…

Game Programming Tools

- Flowcharts: Document algorithms
  - Visio: Basic Flowchart Shapes
  - Process:
    - Rectangle…
  - Decision:
    - Diamond…
  - Start/End:
    - Capsule…
  - Data Input/Output:
    - Parallelogram…



Console Game Programming

- C++ Overview:
  - A Fast Introduction!
    - Given you already know how to program…
      - We can learn C++ and apply it in our exercises and assignments for 717310!
  - Tool Chain:
    - Preprocessor, Compiler, Linker
      - IDE: Visual Studio
  - Procedural Programming with C++…
    - We will introduce OO C++ in a future session…

Console Game Programming

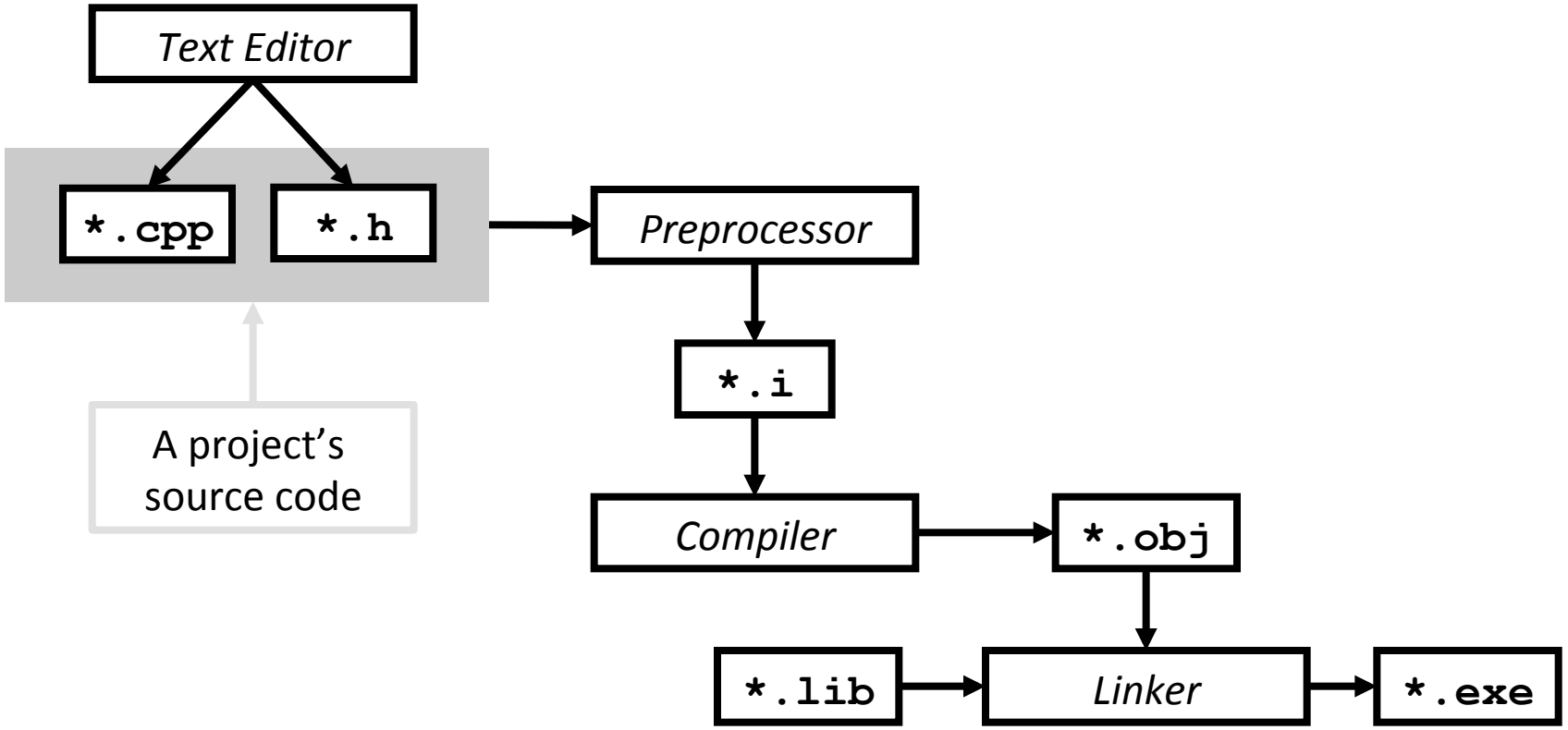
- C++ Overview continued…
  - Today's Procedural Programming Topics:
    - Types, Variables, Console Input/Output, **`sizeof`**
    - Bitwise, Relational and Logical Operators
    - Selection, Repetition
    - Arrays, Enumerations
    - Functions, The Stack,
  - Next time…
    - Pointers, Function Pointers, References
    - Structs, Freestore Allocations, Casting

Console Game Programming

- C++: Tool Chain
  - Build Tools:
    - Preprocessor
    - Compiler
    - Linker
  - Integrated Development Environments:
    - Microsoft Visual Studio:
      - Xbox, Xbox 360, Xbox One
    - SN System's ProDG:
      - PlayStation Vita, PlayStation 3, PlayStation 2, PlayStation Portable, Nintendo DS, GameCube, Game Boy Advance

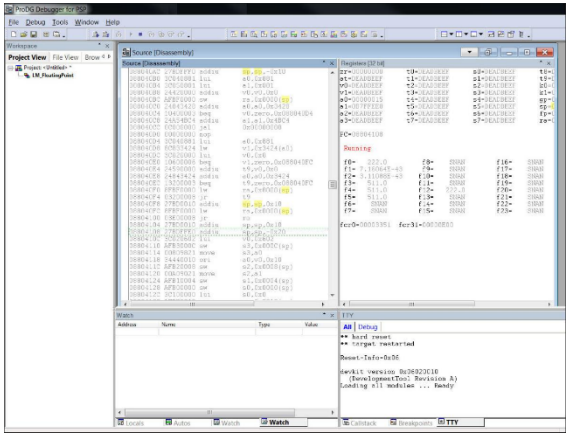
Console Game Programming

- C++: Tool Chain continued…



Console Game Programming

- C++: Tool Chain continued…
  - SN System's ProDG:
    - Console Specific Tools:
      - Assemblers, C/C++ Compiler, ELF/DLL Linkers.
      - Source Code Debugger.
      - Profiler, Performance Optimiser.
      - Target Manager.
    - Command Line Interface
    - Can be integrated with an IDE:
      - Visual Studio
    - Art Tools:
      - ProView



http://img.informer.com/screenshots/2663/2663461_2.jpg

Console Game Programming

- C++: Tool Chain
  - Source Code Files (Plain Text Files):
    - Implementation files: **`*.cpp`**
      - Function bodies…
      - File extensions sometimes: **`.c++`** or **`.cc`**
    - Header files: **`*.h`**
      - Interfaces…
      - Function prototypes / signatures…
      - File extensions sometimes: **`.hpp`** or **`.hh`**
    - Split Architecture…
    - These are the input into Preprocessor…

Console Game Programming

- C++: Example Source File (example.cpp)
  - The simplest C++ program…
  - The C++ entry point is the **`main`** function:

```
int main()
{
    return (0);
}
```

Console Game Programming

- C++: Preprocessor
  - Preprocessor Directives: **#** symbol…
    - **`#include`**
      - Inserts the included file at the include location…
    - **`#define identifier replacement`**
      - Macro Functions…
    - **`#ifdef`**, **`#endif`**, **`#ifndef`**, **`#elif`**, **`#else`**
      - Conditional Compilation…
    - Predefined Macros:
      - **`__LINE__`**, **`__FILE__`**, **`__DATE__`**, **`__TIME__`**
  - From the **`.cpp`** source, the preprocessor creates an intermediate file for the compiler to use…

Console Game Programming

- C++: Tool Chain continued…
  - Translation Units (Compilation unit…):
    - This is the input into the Compiler…
      - It comes from the preprocessor…
- C++: Compiler:
  - Takes preprocessed files as input (translation unit)

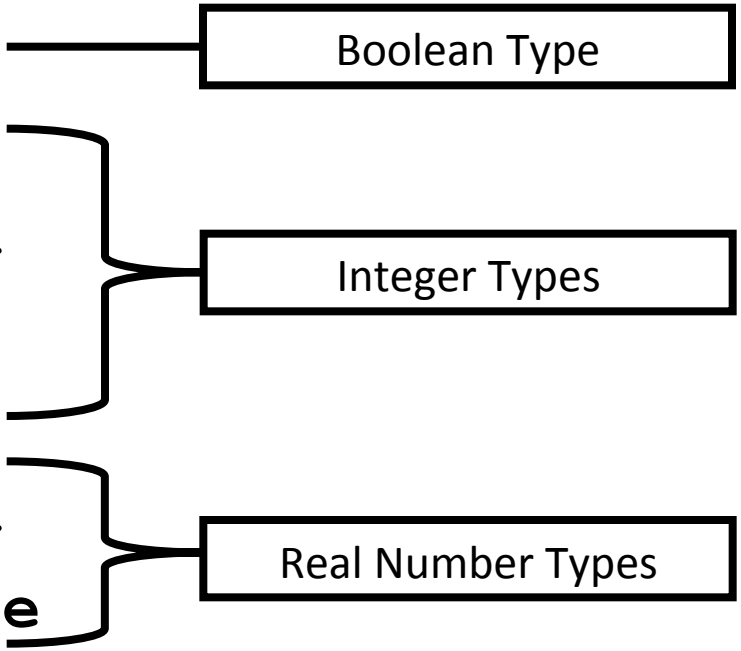
    …
    - Compiler turns the source file (`.cpp`) into an Object Code file: `.obj`
      - Sometimes the file extension is `.o`

Console Game Programming

- C++: Linker
  - Takes all the project's **`.obj`** files as input…
  - Resolves symbols across modules…
  - Generates **`.lib`** files
    - Static library files and headers can be distributed to third parties…
    - Or Dynamic libraries, **`.dll`**
  - Or executables… (for a specific platform…)
    - **`.exe`**, **`.elf`**, **`.xbe`**, etc

Console Game Programming

- C++: Fundamental Data Types
  - Primitive/Intrinsic Types…
    - **`bool`**
    - **`char`**
    - **`short`**
    - **`int`**
    - **`long`**
    - **`float`**
    - **`double`**
    - **`void`**



Console Game Programming

- C++: Fundamental Data Types
  - Types Literals/Constants:
    - bool: **`true`**, **`false`**
    - char: **`'a'`**, **`'A'`**, **`'z'`**, **`'Z'`**, **`'0'`**, **`'9'`**, **`'\n'`**, **`'\''`**
    - short:
    - int:
    - long: **`100L`**
    - float: **`1.0f`**, **`3.14f`**
    - double: **`0.0`**, **`1.0`**, **`3.14`**
    - void

Console Game Programming

- C++: Using Variables
  - Declaring, Initialising:

```
int main()
{
    int i = 10;
    bool b = true;
    float f = 3.14f;
    char c = 'a';
    return (0);
}
```

Console Game Programming

- C++: Using Variables continued…
  - Arithmetic Operators: **+**, **-**, *****, **/**, **%**

```
int main()
{
    int i = 10;
    i = ((i * 5) + 3);
    ++i; // Pre-increment.
    i++; // Post-increment.
    i += 5;
    //...
```

Console Game Programming

- C++: Printing to the Console
  - C++ Standard Library:

    ```
    #include <iostream>
    ```

    - Input / Output Stream…
  - Printing to the Console:

    ```
    std::cout << "Hello" << std::endl;
    ```
  - Reading from the Console:

    ```
    int i = 0;
    std::cin >> i;
    ```

Console Game Programming

- C++: Using Variables continued…
  - Keyword: **`sizeof`**: Returns the number of bytes used by a type or variable…
  - For example:

```
int main()
{
    std::cout << "An int is ";
    std::cout << sizeof(int);
    std::cout << " bytes" << std::endl;
    return (0);
}
```

Console Game Programming

- C++: Bitwise Operators
  - Bitwise AND: **&**
  - Bitwise OR: **|**
  - Bitwise XOR: **^**
  - Bitwise NOT: **~**
  - Bitwise Shift Left: **<<**
  - Bitwise Shift Right: **>>**
  - Example:
    - `int r = ((0x20 | 0x10) & 0x1F) << 2;`

Console Game Programming

- C++: Relational Operators
  - Equality: **==**
  - Not equal: **!=**
  - Greater than: **>**
  - Greater than or equal to: **>=**
  - Less than: **<**
  - Less than or equal to: **<=**
  - Evaluate to become a **`bool`**.
    - **`true`** or **`false`**.

Console Game Programming

- C++: Logical Operators
  - Logical AND: **`&&`**
  - Logical OR: **`||`**
  - Logical NOT: **`!`**
  - Evaluate to become a **`bool`**.
    - **`true`** or **`false`**.
  - Example:

```
int x = 50;
int y = -25;
bool b = x > 5 && y < 6;
```

Console Game Programming

- C++: Selection
  - Keywords: **`if`**, **`else`**
  - Example:

```
if (x > 10)
{
    // x is greater than 10
}
else if (x > 5)
{
    // x is greater than 5, but less than or equal to 10
}
else
{
   // x is less than or equal to 5.
}
```

Console Game Programming

- C++: Repetition
  - Keywords: **`for`**, **`while`**, **`do`**
  - For Example:

```
for (int i = 0; i < 5; ++i)
{
    std::cout << "i is: " << i << std::endl;
}
```

  - While Example:

```
int i = 0;
while (i < 5)
{
    std::cout << "i is: " << i << std::endl;
    ++i;
}
```

Console Game Programming

- C++: Repetition continued…
  - Keywords: **`for`**, **`while`**, **`do`**
  - Do While Example:

```
int i = 0;
do
{
    std::cout << "i is: " << i << std::endl;
    ++i;
}
while (i < 5);
```

  - The do while loop will execute the body of the loop once… whereas the while loop may never execute the body… depending on the condition…

Console Game Programming

- C++: Arrays
  - Contiguous blocks of memory…
  - Example:

```
int i[5];
i[0] = -1;
i[1] = 2;
i[2] = -3;
i[3] = 4;
i[4] = -5;
```

  - Beware! Out of bounds will compile and run!

```
i[5] = 10;
i[-1] = 20;
```

  - Memory trampling at runtime!

Console Game Programming

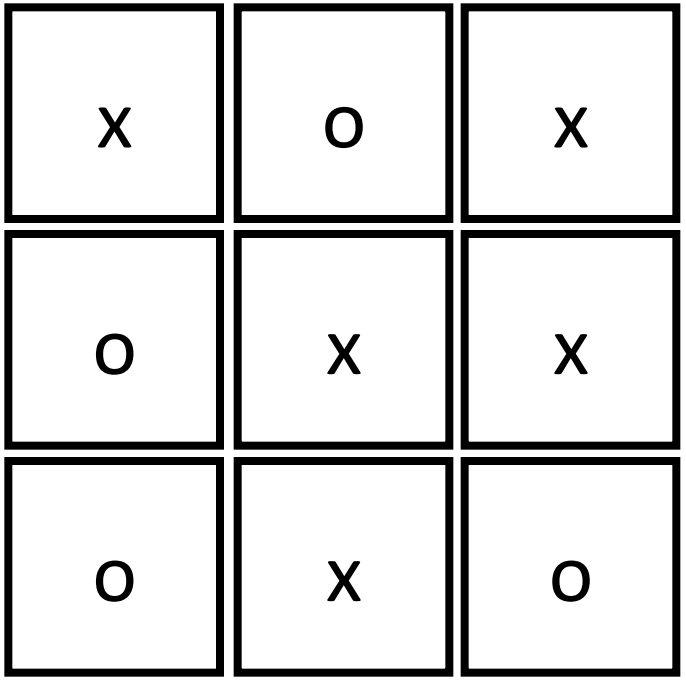
- C++: Character Arrays
  - Character Literals: **`'x'`**
    - Single Byte Value. ASCII…
  - String Literals: **`"Hello 717310!"`**
    - Null character terminated…
  - Example:

```
char example1[] = { 'H', 'e', 'l', 'l', 'o', '\0' };
char example2[] = "Hello";
example2[0] = 'n';
example2[1] = 'e';
example2[2] = 'w';
example2[3] = '\0';
```

Console Game Programming

- C++: Multi-Dimensional Arrays
  - Contiguous blocks of memory…
  - Example:

```
char board[3][3];
board[0][0] = 'x';
board[0][1] = 'o';
board[0][2] = 'x';
board[1][0] = 'o';
board[1][1] = 'x';
board[1][2] = 'x';
board[2][0] = 'o';
board[2][1] = 'x';
board[2][2] = 'o';
```



Console Game Programming

- C++: Function Declaration
  - The signature… or prototype…
  - For example:

```
void func(int x, int y);
```

- C++ Function Definition
  - The body… function implementation…
  - For example:

```
void func(int x, int y)
{
    std::cout << "x is " << x << std::endl;
}
```

Console Game Programming

- C++: The Stack
  - The Call Stack:
    - An area of memory, used for storing information about the active functions used by the program.
    - At runtime:
      - Function called: Pushed onto the stack.
      - Function returns: Popped off the stack.
  - Each Stack Frame (Activation Frame) stores:
    - Parameters, Local Variables, Return Address.
    - For the called function…

Console Game Programming

- C++: Enumeration
  - A user type whose values are explicit constants…
  - Example:

```
enum MyType
{
    VALUE_A,    // Defaults to integer value of 0.
    VALUE_B,    // Defaults to integer value of 1.
    VALUE_C     // Defaults to integer value of 2.
};

int main()
{
   MyType var = VALUE_B;
   //...
```

Console Game Programming

- C++: Structures
  - Hold data… Create complex data types…
    - Member data, fields, properties…
  - Keyword: **`struct`**
  - Example Struct Declaration:

```
struct Person
{
    int age;
    int weight;
};
```

Console Game Programming

- C++: Using Structures
  - Example:

```
// The struct is declared above here...
int main()
{
    Person me;
    me.age = 40;
    me.weight = 100;

    std::cout << me.age << std::endl;
    std::cout << me.weight << std::endl;
}
```

Console Game Programming

Precedence	Operator	Description	Associativity
1	`::`	Scope Resolution	Left-to-right

2	`++ --`
	`type() type{}`
	`()`
	`[]`
	`.`
	`->`
		Suffix/postfix increment and decrement
		Function-style type cast
		Function call
		Array subscripting
		Element selection by reference
		Element selection through pointer
			Left-to-right

3	`++ --`
	`+ -`
	`! ~`
	`(type)`
	`*`
	`&`
	`sizeof`
	`new, new[]`
	`delete, delete[]`
		Prefix increment and decrement
		Unary plus and minus
		Logical NOT and bitwise NOT
		C-Style type cast
		Indirect (dereference)
		Address-of
		Size-of
		Dynamic memory allocation
		Dynamic memory deallocation
			Right-to-left

4	`.* ->*`	Pointer to member	Left-to-right
5	`* / %`	Multiplication, division and remainder	Left-to-right
6	`+ -`	Addition and subtraction	Left-to-right
7	`<< >>`	Bitwise left shift and right shift	Left-to-right

Console Game Programming

Precedence	Operator	Description	Associativity
8	`< <=` `> >=`	Relational operators, less than, or equal to Relational operators, greater than, or equal to	Left-to-right
9	`== !=`	Relational operator, equal to, not equal	Left-to-right
10	`&`	Bitwise AND	Left-to-right
11	`^`	Bitwise NOR (Exclusive OR)	Left-to-right
12	`|`	Bitwise OR (Inclusive OR)	Left-to-right
13	`&&`	Logical AND	Left-to-right
14	`||`	Logical OR	Left-to-right

15	`?:`
	`=`
	`+= -=`
	`*= /= %/`
	`<<= >>=`
	`&= ^= |=`
		Ternary Condition
		Direct Assignment
		Assignment by sum and difference
		Assignment by product, quotient, remainder
		Assignment by bitwise left shift and right shift
		Assignment by bitwise AND, XOR, and OR
			Right-to-left

16	`Throw`	Throw Operator	Right-to-left
17	`,`	Comma	Left-to-right

Precedence and associativity are independent from order of evaluation.

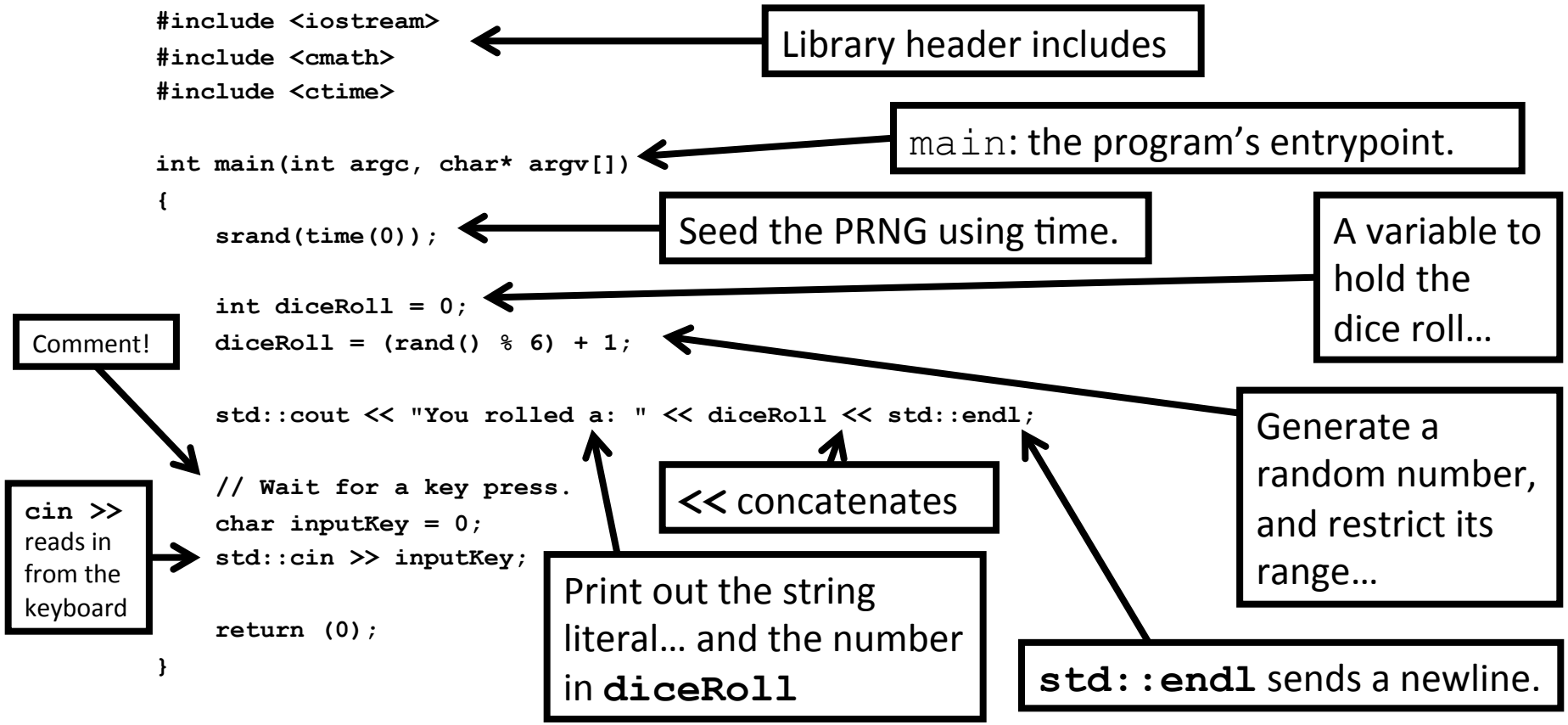
Game Programming Tools and Libraries

- Simple random number generation in C/C++:

```
#include <iostream>
#include <cmath>
#include <ctime>

int main(int argc, char* argv[])
{
    srand(time(0));

    int diceRoll = 0;
    diceRoll = (rand() % 6) + 1;

    std::cout << "You rolled a: " << diceRoll << std::endl;

    // Wait for a key press.
    char inputKey = 0;
    std::cin >> inputKey;

    return (0);
}
```

Game Programming Tools and Libraries

- Simple random number generation in C/C++:



Exercises

- Week 1:
  - Day 001.1 – “Simple” Dice Game
  - Day 001.2 – Noughts and Crosses
  - Day 001.3 – “Simple” Dice Game, with Statistics Reporting

Exercise: “Simple” Dice Game

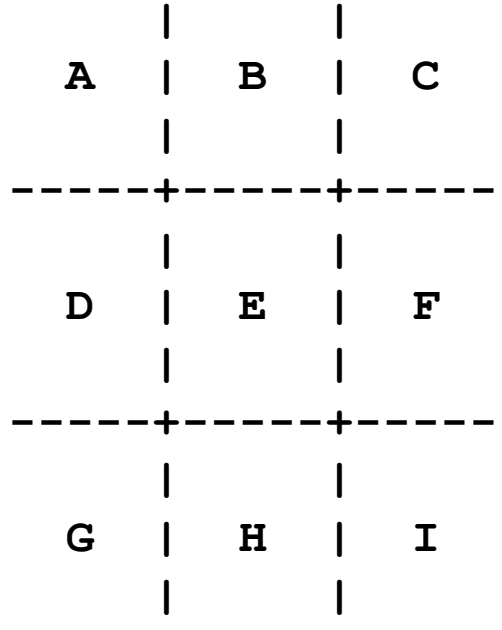
- Implement a simple game of dice in C++:
  - Two players: one human, one computer AI.
  - The rules:
    - Each player gets one turn per round.
    - On a turn the player rolls two six-sided dice.
    - The player that gets a double (or highest double) wins the round.
    - If neither player gets a double, the highest total wins.
    - If both players roll the same total, then it’s a draw.
    - On the next round, the player swap turns.
      - Previous second roller is now the first roller… etc.

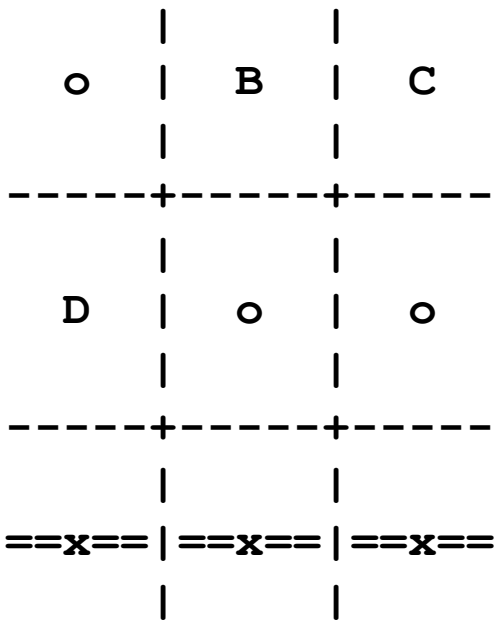
Exercise: “Simple” Dice Game

- Implement a simple game of dice continued…
  - The game rounds continue, until the human player decides to quit.
  - There is one final requirement…
    - The computer AI must have a winning average of close to 70%.
    - However, the game should still appear random…
      - And hence fair to the player…
  - Start by creating a flowchart to document the logic for the game design…
    - Then implement a text-based version of game…

Exercise: Noughts and Crosses

- Implement a simple game of Noughts and Crosses in C++:
  - Display the boards as follows:





Exercises

- Recommended Readings:
  - Harris, L. (2014). *New Zealand at a Glance*. Retrieved from http://www.develop-online.net/news/new-zealand-at-a-glance/0195245
  - Batchelor, J. (2014). *DayZ creator Dean Hall reveals his post-Bohemia plans*. Retrieved from http://www.develop-online.net/news/dayz-creator-dean-hall-reveals-his-post-bohemia-plans/0194269

Exercises

- Recommended Reference Books:
  - Rabin, S. (2010). *Introduction to Game Development* (2nd ed.). Portland, OR: Cengage Learning.
  - Zackariazzon, P. (2012). *The Video Game Industry: Formation, Present State, and Future*. New York, NY: Routledge.

Exercises

- Recommended Readings:
  - Dawson, M. (2010). *Beginning C++ Through Game Programming* (3rd ed.). Boston, MA: Cengage Learning PTR.
  - Stroustrup, B. (2013). *The C++ Programming Language* (4th ed.). Upper Saddle River, NJ: Addison-Wesley Professional.
  - Llopis, N. (2003). *C++ for Game Programmers*. Hingman, MA: Charles River Media, Inc.

Summary

- Procedural C++
  - An introduction to Programming in C++
- Exercises