C++ Basics

```
// A simple program
#include <iostream>
using namespace std;

int main() {

    cout << "Programming is fun.\n";
    return 0;

}
```

People read code.
Use comments so
a human can
understand why
and how

C++ Basics

```
// A simple C++ program
# include <iostream>
using namespace std;

int main() {

    cout << "Programming is fun.\n";
    return 0;

}
```

Pre proccessor instructions come before "main"

C++ Basics

```
// A simple C++ program
#include <iostream>
using namespace std;

int main() {

    cout << "Programming is fun.\n";
    return 0;

}
```

the start of main

the end of main
- its the curly brace "}" that matches

{ and } mark off code blocks

Every code block starts with { and ends with } and Contains 0 or more statetements and code blocks (and nothing else)

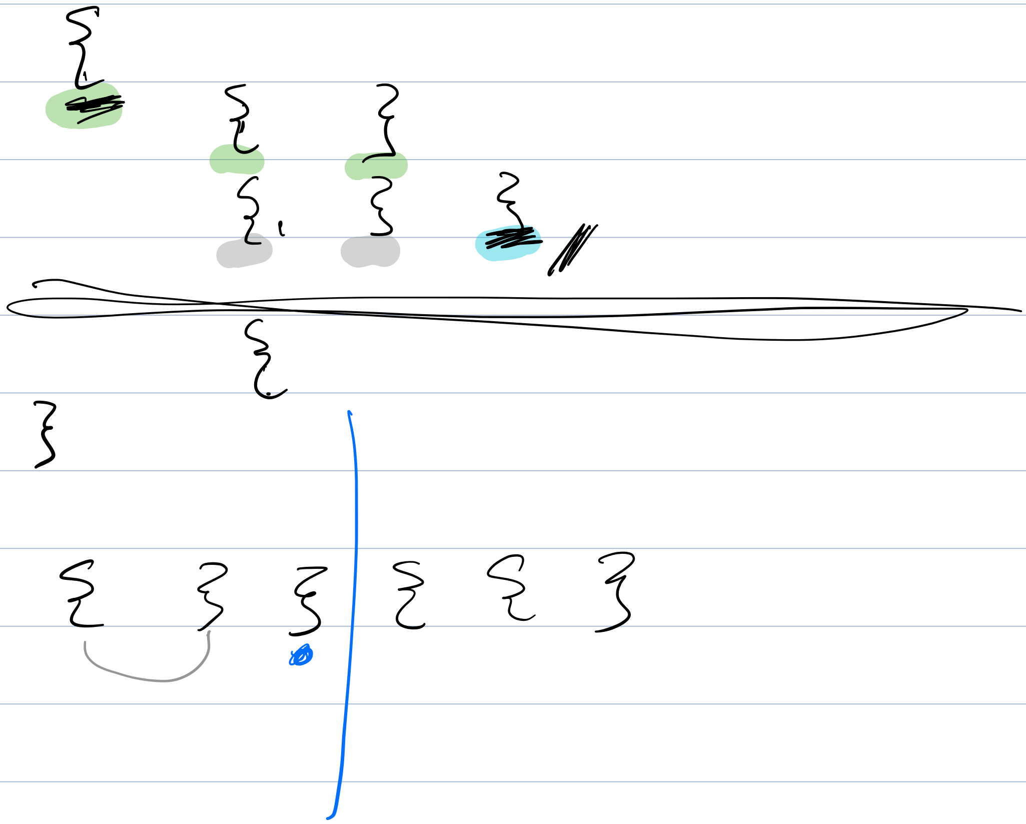
Is this a code block?

```
  {




}
```

Is this a code block?



C++ Basics

```
// A simple C++ program
#include <iostream>
using namespace std;

int main() {

    cout << "Programming is fun.\n";
    return 0;

}
```

The code inside main.

- statements
- code blocks deliminated by { }

C++ Statements

Examples

```
int a, b, c;

float average;
```

In olden times these had to go at the top, before any other types of statements in the block. Modern C++ lets you put them anywhere a statement can go.

"W" warning if you define a var that you don't use.

C++ Statements

Examples

```
int a, b, c;

float average;
```

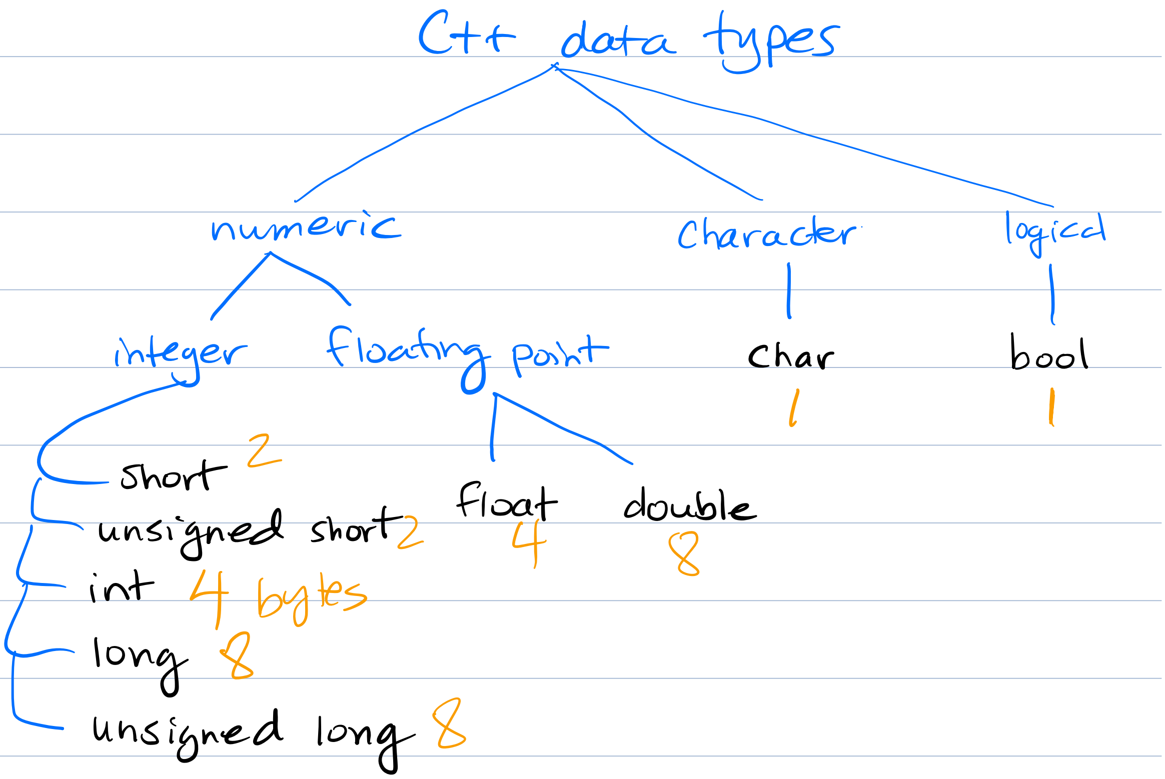
Variable names:

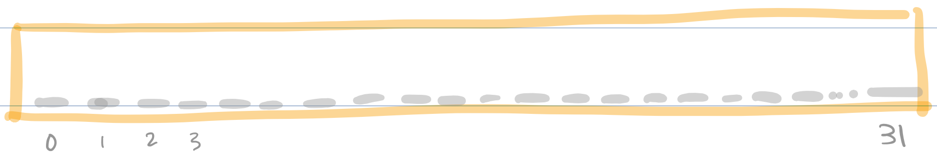
- start with [a..z] + [A..Z] + _
- after that, 0 or more symbols of same kind + [0..9]
- case-sensitive

Eg - which of these are legal var names and are any two equivalent (refer to same "space" for data)?

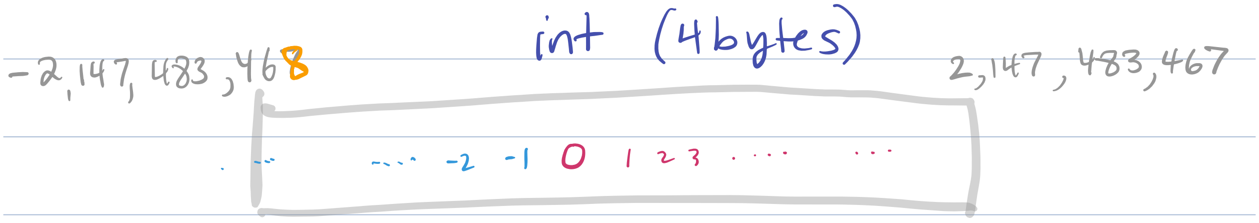
dayOfWeek ✓
dayofweek ✓
_day_1997 ✓
1997day ✗

_mix #3 X

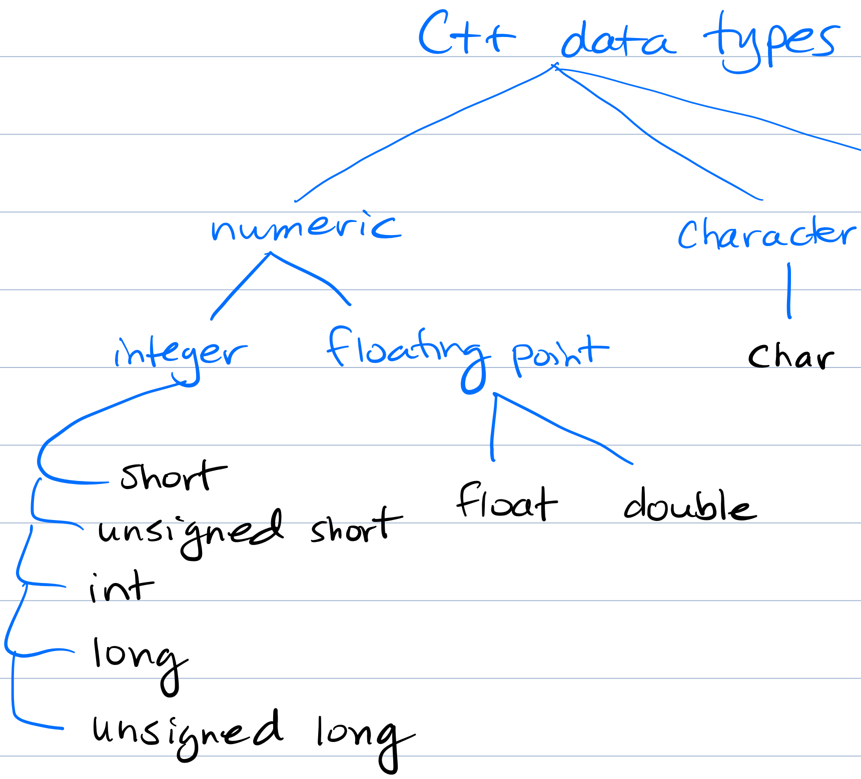


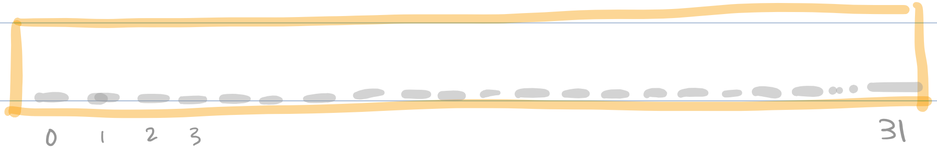


There are 2^{32} different bitstrings of length 32 - ie different numbers that can be encoded with 4 bytes (8 bits per byte)



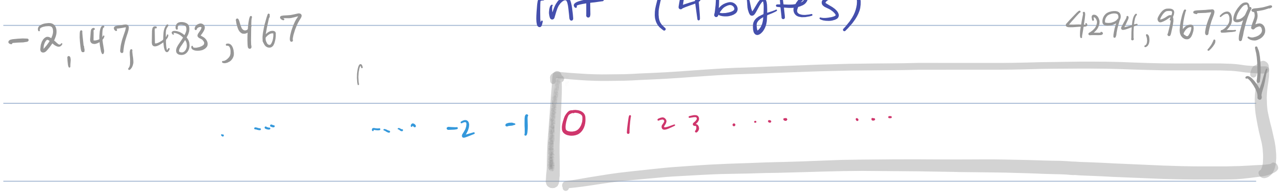
_mix #3





There are 2^{32} different bitstrings of length 32 – ie different numbers that can be encoded with 4 bytes (8 bits per byte)

unsigned int (4 bytes)



Integer Type	bytes	Range
short	2	-32,768 - +32,767
unsigned short		0 - +65,535
int		-2,147,483,648 - +2,147,483,647
unsigned int		0 - +4,294,917,295
long		-
unsigned long		
long long		

Integer Type		Range		
short	2	SHRT_MIN	-	SHRT_MAX
unsigned short		0	-	USHRT_MAX
int		INT_MIN	-	INT_MAX
unsigned int		0	-	UINT_MAX
long		LONG_MIN	-	LONG_MAX
unsigned long		0	-	ULONG_MAX
long long		LLONG_MIN	-	LLONG_MAX

This limits are defined in <limits.h>

Note: Modern C++ : <climits>

Floating Point data types

For use when real numbers (actually rationals) are to be represented.

Decimal	Scientific	E-notation
247.91	2.47×10^{2}	
0.00072	7.2×10^{-4}	
2,900,000	2.9×10^{6}	

Floating Point data types

For use when real numbers (actually rationals) are to be represented.

Decimal	Scientific	E-notation
247.91	2.4791×10^{2}	2.4791E2
0.00072	7.2×10^{-4}	7.2E-4
2,900,000	2.9×10^{6}	2.9E6

C++ data types for floating point numbers

					Significant digits
float	4 bytes	±3.4E-38	–	±3.4E+38	7
double	8 bytes	±1.7E-308	-	±1.7E30	16
long double	8 bytes	"		"	16

When writing a literal floating point number there are a variety of ways:

```
double a;
a = 1.496E8;
a = 149600000;
```

Assigning floating point values to int vars:

```
int number;
number = 1.7;
// number now has value 1
// because c++ truncates the fractional part.

int intVar;
double doubleVar = 7.8;
intVar = doubleVar; // assigns 7 to intVar
```

Careful! double has more bytes than int
So the truncated value may be outside the
range of int! Value may be invalid.

The char data type

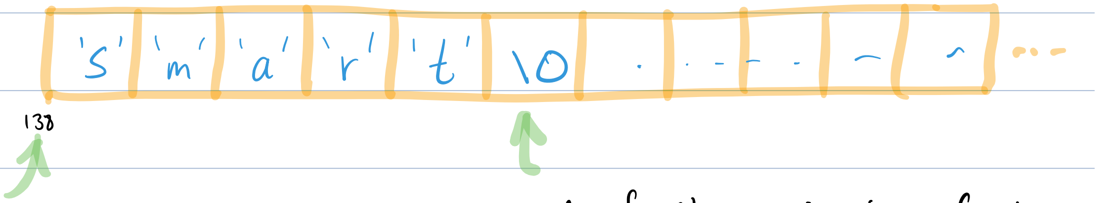
char type variables hold a single character
from the alphabet

- with 8 bits you can encode 256 different characters

'A'	'B'	'C'	...	'a'	
65	66	67	...	9	← ASCII encoding

This is called the ASCII encoding
See cppreference, search ASCII

We will be using a complex data type called a string which is a sequence of chars stored consecutively in memory ending with the char called Null whose ASCII encoding is \0 (value 0, not symbol '0')



Start of string is address of first char

end of string is the first zero-valued entry

C++ automatically puts a null terminator at the end of string constants.

```
cout << "Hello, world!" << endl;
```

In memory, would be

'H'	'e'	'l'	'l'	'o'	','	'␣'	'W'	'o'	'r'	'l'	'd'	'!'	0

Null
also written \0

blank (' ') is also a character - character 32

Literals: not a named thing, but are given literally

1.23 - eg x = 1.23

"A" - is stored as a string

65	0

'A' - is stored as a char

65

```
char letter = "A"; // won't work
```

C++ type bool

bool type variables have value either
true or false
|| ||
1   0

```
bool    isRaining = true;
        if  (isRaining)  {
              cout << "Wear a raincoat!" <<endl;
        }
```