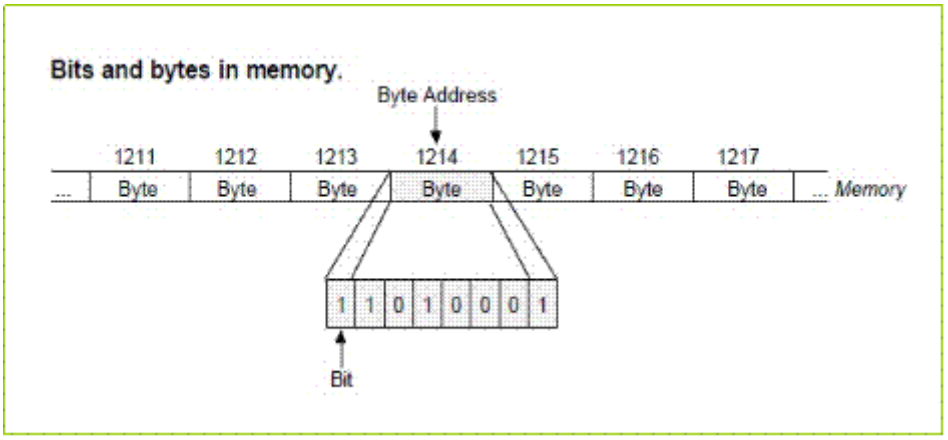
Lesson 3

Memory

Memory is simply an ordered list of bits. A bit is the smallest amount of data a computer can store and is represented by either a 1 or a 0.
These are grouped into bytes (8 bits in a byte) and numbered.
This number is the memory location or address.
In the example below that would be 1214.



Pointers

Pointers are variables that hold these addresses.

- They point to values stored in memory
- They can be manipulated like regular variables

Pointers have a data type the same as the data they point to.

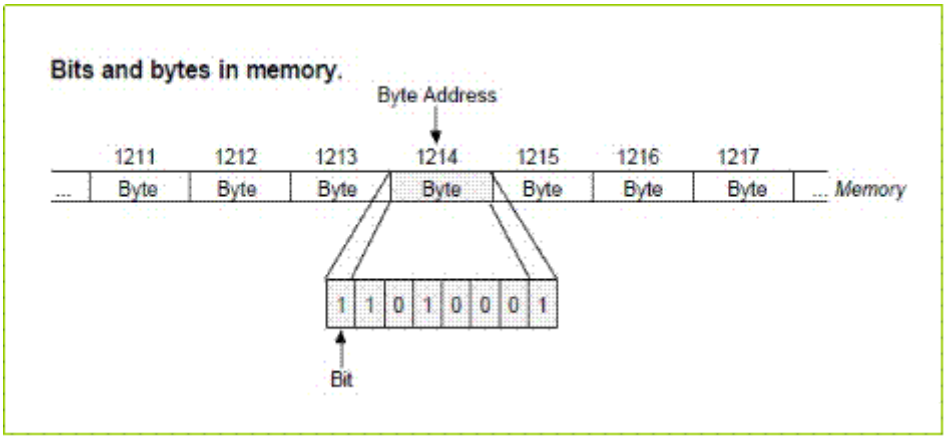
```
int main() {
  int aNumber = 10;
  int *aPointer;       //A pointer to an integer
  aPointer = &aNumber; //aPointer is assigned the address of aNumber
}
```

```
cout << aNumber << endl;
cout << aPointer << endl;  //Memory address
cout << *aPointer << endl; //Contents of memory address
```

```
10
0x72fe44
10
```

Using the image below as an example

- aNumber would equal 11010001
- aPointer would equal 1214



More About Pointers

In the previous example we saw the use of the &. For a normal variable using the & gives the memory address of where that variable is stored. This is called dereferencing. The following example will print out the memory address of where *aNumber* is stored.

```
int main() {
  int aNumber = 10;
  cout << &aNumber << endl;
}
```

When I ran the program I found that *aNumber* was stored at the memory address 0x72fe4c.

```
C:\Users\Daire\Documents\GitHub\cppDayCourse\Lesson3>.\PointerTest
0x72fe4c
```

We can also dereference a pointer to see where in memory it is stored.

```
int main() {
  int aNumber = 10;
  int *aPointer;
  aPointer = &aNumber;
  cout << &aNumber << endl;
  cout << &aPointer << endl;
}
```

The memory addresses where aNumber and aPointer are stored.

```
0x72fe4c
0x72fe40
```

Because aPointer points to aNumber the memory address of aNumber will be the same as the value of aPointer. So when we run the code below the same memory address will be printed twice. The code and result can be seen below.

```
cout << &aNumber << endl;
cout << aPointer << endl;
```

```
0x72fe44
0x72fe44
```

Similarly the below code will print out the same thing twice. That being the value of aNumber.

```
cout << aNumber << endl;
cout << *aPointer << endl;
```

```
10
10
```

Tasks

- Create two integers *a* and *b*
  - Set *a* = 10
  - Set *b* = 20
- Create two pointers to integers
- Let the first pointer point at *a* and the second point at *b*
- Print out the value that each pointer points to
- Use the pointers to change the value of
  - *a* to 15
  - *b* to 25
- Print out the value that each pointer points to
- Let the second pointer point at *a* and the first pointer point at *b*
- Print out the value that each pointer points to

The result should look something like this

```
The first pointers value is 10
The second pointers value is 20
The values the pointers point to has been changed
The first pointers value is 15
The second pointers value is 25
The pointers now point to new values
The first pointers value is 25
The second pointers value is 15
```

N.B. Make sure you are doing this using pointers