Intro

This tutorial provides a brief introduction to C++ data types.

What is a data type?

When we wish to store data in a C++ program, such as a whole number or a character, we have to tell the compiler which type of data we want to store. The data type will have characteristics such as the range of values that can be stored and the operations that can be performed on variables of that type.

Fundamental types

C++ provides the following fundamental built-in data types: Boolean, character, integer and floating-point. It also enables us to create our own user-defined data types using enumerations and classes.

For each of the fundamental data types the range of values and the operations that can be performed on variables of that data type are determined by the compiler. Each compiler should provide the same operations for a particular data type but the range of values may vary between different compilers.

Boolean Type

The Boolean type can have the value true or false. For example:
```
bool isEven = false;
bool keyFound = true;
```

If a Boolean value is converted to an integer value true becomes 1 and false becomes 0.

If an integer value is converted to a Boolean value 0 becomes false and non-zero becomes true.

Character Type

The character type is used to store characters - typically ASCII characters but not always. For example:
```
char menuSelection = 'q';
char userInput = '3';
```

Note how a character is enclosed within single quotes. We can also assign numeric values to variables of character type:
```
char chNumber = 26;
```

We can declare signed and unsigned characters, where signed characters can have positive and negative values, and unsigned characters can only contain positive values.
```
signed char myChar = 100;
signed char newChar = -43;
unsigned char yourChar = 200;
```

Note that if we use a plain char, neither signed nor unsigned:
char dataValue = 27;

it may differ between compilers as to whether it behaves as a signed or unsigned character type. On some compilers it may accept positive and negative values, on others it may only accept positive values. Refer to your compiler documentation to see which applies.

A char is guaranteed to be at least 8 bits in size. C++ also provides the data type wchar_t, a wide character type typically used for large character sets.

An array of characters can be used to contain a C-style string in C++. For example:
char aString[] = "This is a C-style string";

Note that C++ also provides a string class that has advantages over the use of character arrays.

Integer Types

The integer type is used for storing whole numbers. We can use signed, unsigned or plain integer values as follows:
signed int index = 41982;
signed int temperature = -32;
unsigned int count = 0;
int height = 100;
int balance = -67;

Like characters, signed integers can hold positive or negative values, and unsigned integers can hold only positive values. However, plain integer can always hold positive or negative values, they're always signed.

You can declare signed and unsigned integer values in a shortened form, without the int keyword:
signed index = 41982;
unsigned count = 0;

Integer values come in three sizes, plain int, short int and long int.
int normal = 1000;
short int smallValue = 100;
long int bigValue = 10000;

The range of values for these types will be defined by your compiler. Typically a plain int can hold a greater range than a short int, a long int can hold a greater range than a plain int, although this may not always be true. What we can be sure of is that plain int will be at least as big as short int and may be greater, and long int will be at least as big as plain int and may be greater. A short integer is guaranteed to be at least 16 bits and a long integer at least 32 bits.

You can declare short and long integer values in a shortened form, without the int keyword:
short smallValue = 100;
long bigValue = 10000;

You can have long and short signed and unsigned integers, for example:

```
unsigned long bigPositiveValue = 12345;
signed short smallSignedValue= -7;
```

Floating-Point Types

Floating point types can contain decimal numbers, for example 1.23, -.087. There are three sizes, float (single-precision), double (double-precision) and long double (extended-precision). Some examples:

```
float celsius = 37.623;
double fahrenheit = 98.415;
long double accountBalance = 1897.23;
```

The range of values that can be stored in each of these is defined by your compiler. Typically double will hold a greater range than float and long double will hold a greater range than double but this may not always be true. However, we can be sure that double will be at least as great as float and may be greater, and long double will be at least as great as double and may be greater.

Enumeration Type

An enumeration type is a user defined type that enables the user to define the range of values for the type. Named constants are used to represent the values of an enumeration, for example:

```
enum weekday {monday, tuesday, wednesday, thursday, friday, saturday, sunday};
weekday currentDay = wednesday;
if(currentDay==tuesday){
// do something
}
```

The default values assigned to the enumeration constants are zero-based, so in our example above monday == 0, tuesday == 1, and so on.

The user can assign a different value to one or more of the enumeration constants, and subsequent values that are not assigned a value will be incremented. For example:

```
enum fruit {apple=3, banana=7, orange, kiwi};
```

Here, orange will have the value 8 and kiwi 9.

Class Type

The class type enables us to create sophisticated user defined types. We provide data items for the class type and the operations that can be performed on the data. For example, to create a square class that has a data item for size, and provides draw and resize operations:

```
class square {
int size;
public:
square();
~square();
void draw();
bool resize(int newSize);
};
```

[Please refer to a tutorial on classes and objects for a more detailed explanation of the class type, which is outside the scope of this tutorial].

Bob @ http://www.daniweb.com/techtalkforums/thread1767.html