Beginners Programming Course

for the

Glasgow Electronics Club.

by

Dave Ball

Table of Contents

Introduction

The purpose of this course is as an introduction to the concepts, terminology and practices of computer programming and assumes no prior knowledge of computer programming. To do this the course will concentrate on two programmming laungues C++ and PHP. Why two lanuages you may ask? by teach two launagues i hope to enable those on the course to more easily understand the concepts, but also show them how the same concept can be implement in an differnet launagues enabling the student to then use these concept in any programming laungue. All sample code will be written and destributed as netbeans projects and xampp has been used for the PHP environment.

Setting up your environment

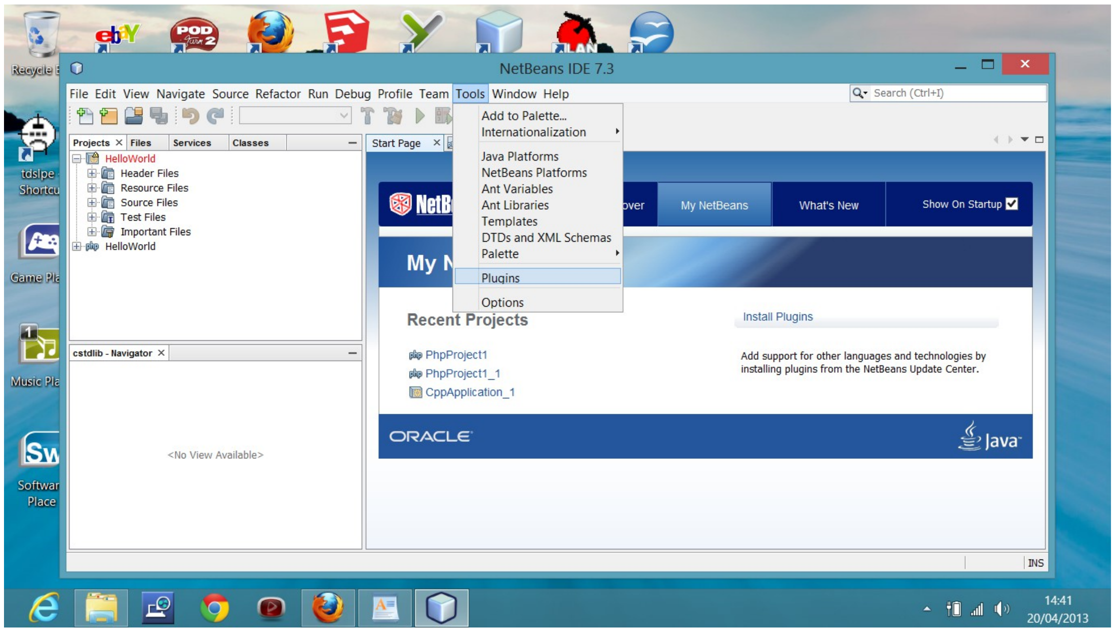
In order to write a computer program you need certain applications installed, now in theory you can write a computer program in any text editor, but in doing so you are just making life hard for your self, now there are many text editors avalible and some like notepad++ and other will even colour code and check the syntax of your code. But in writing using just a text editor you deprive yourself of many for the tools that are avaible in and IDE(Intergarted Developement Evironment). For this course we will be using an IDE called netbeans. This is an open source IDE and supports via plugins many programming launauges including C++ and PHP.

Go to https://netbeans.org/ and select the full application downlaod and install it.

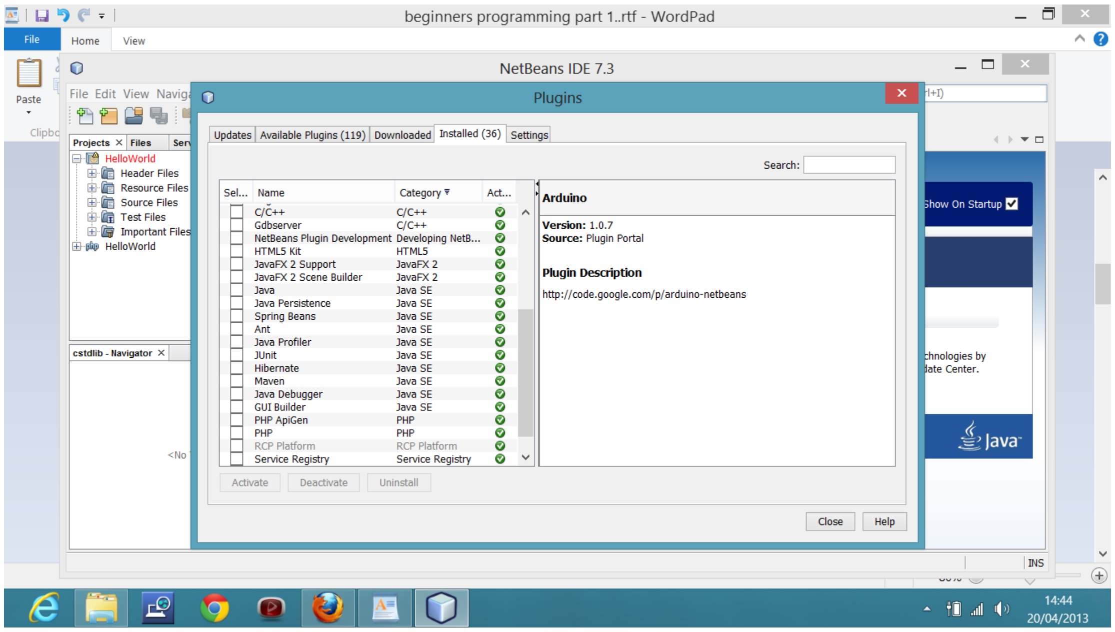
{ will fill in step by step instructions with screen shots later netbeans appears be be down for maintanice at the moment}

Now for the PHP enviroment you will also need to install http://sourceforge.net/projects/xampp/ . XAMPP is a very easy to install Apache Distribution for Linux, Solaris, Windows and Mac OS X. The package includes the Apache web server, MySQL, PHP, Perl, a FTP server and phpMyAdmin. **Caution you do not need to carry out this task if you already have the packages mentioned above installed.** If youare only interested in learning C++ please install Xampp as we will be using some of the application later in the course.

Once these are install you will need to check the pluggins section of netbeans to make sure you have the pluggins for PHP and C++ installed. Select Tools and then Plugins from the Menu,



now you want to check that the C++ and PHP plugin as installed.



Sel...	Name	Category	Act...
☐	C/C++	C/C++	✓
☐	Gdbserver	C/C++	✓
☐	NetBeans Plugin Development	Developing NetB...	✓
☐	HTML5 Kit	HTML5	✓
☐	JavaFX 2 Support	JavaFX 2	✓
☐	JavaFX 2 Scene Builder	JavaFX 2	✓
☐	Java	Java SE	✓
☐	Java Persistence	Java SE	✓
☐	Spring Beans	Java SE	✓
☐	Ant	Java SE	✓
☐	Java Profiler	Java SE	✓
☐	JUnit	Java SE	✓
☐	Hibernate	Java SE	✓
☐	Maven	Java SE	✓
☐	Java Debugger	Java SE	✓
☐	GUI Builder	Java SE	✓
☐	PHP ApiGen	PHP	✓
☐	PHP	PHP	✓
☐	RCP Platform	RCP Platform	✓
☐	Service Registry	Service Registry	✓

Arduino

Version: 1.0.7
Source: Plugin Portal

Plugin Description

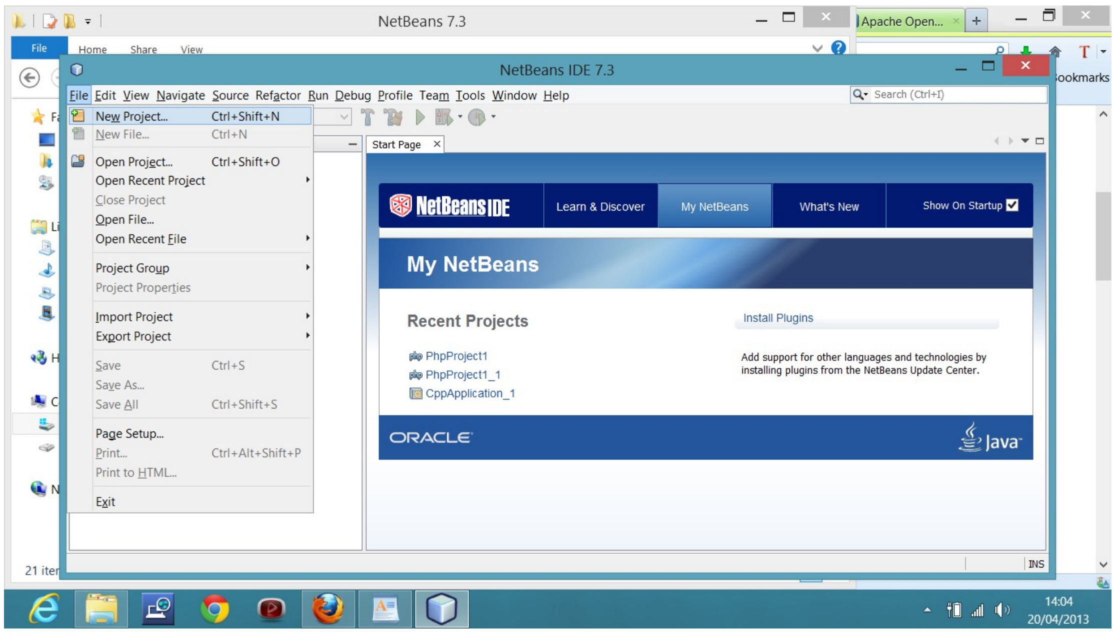
http://code.google.com/p/arduino-netbeans

if they are not select them form the alaibled plugins tab and install them. Your developemt environment is now ready.

Getting Started

Now that you have your development environment installed. We are now in a possition to write your first program. So rather than break with tradition we will use the standard example program used by everyone as there first program. The Hello World example.

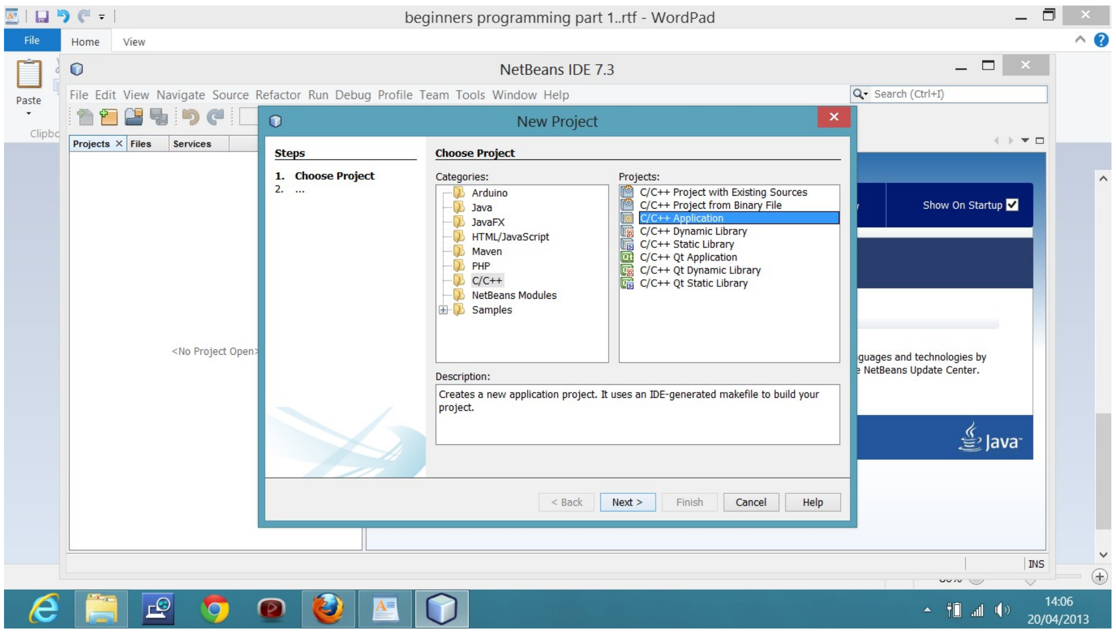
Now open NetBeans.



Select New Project

you then will need to select the type of project you want to write.

If you are writting a C++ application select C/C++ application as shown below.


beginners programming part 1..rtf - WordPad
NetBeans IDE 7.3
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
New Project
Steps
1. Choose Project
2. ...
Back
Next >
Finish
Cancel
Help

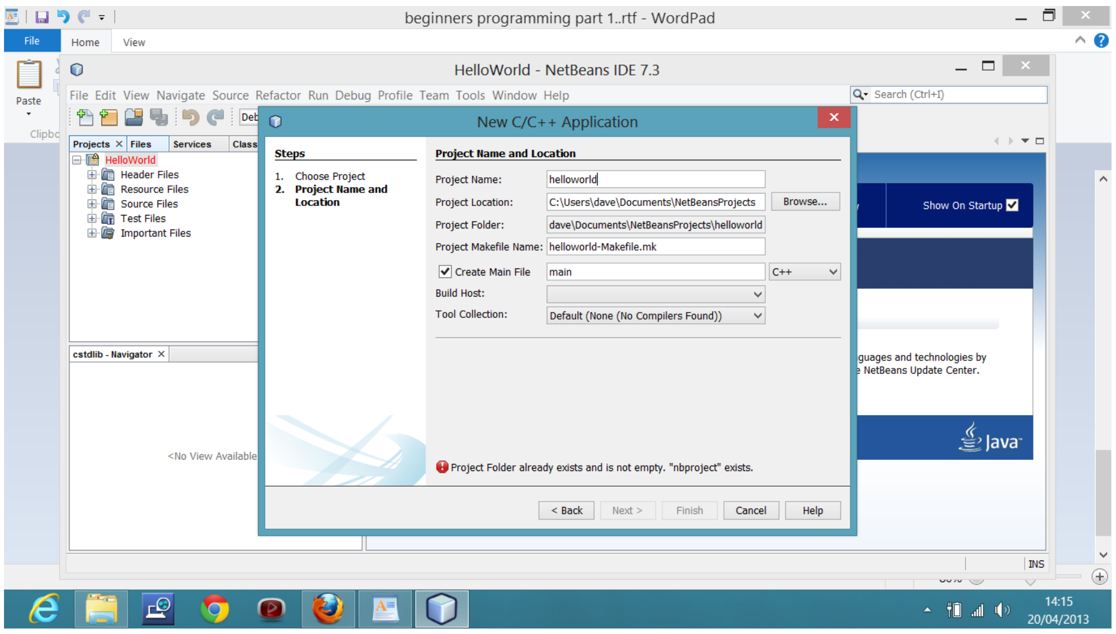
Choose Project

Categories: Arduino, Java, JavaFX, HTML/JavaScript, Maven, PHP, C/C++, NetBeans Modules, Samples

Projects: C/C++ Project with Existing Sources, C/C++ Project from Binary File, C/C++ Application, C/C++ Dynamic Library, C/C++ Static Library, C/C++ Qt Application, C/C++ Qt Dynamic Library, C/C++ Qt Static Library

Description:
Creates a new application project. It uses an IDE-generated makefile to build your project.

you will now be prompted to name your project. Type in HelloWorld into the project name.

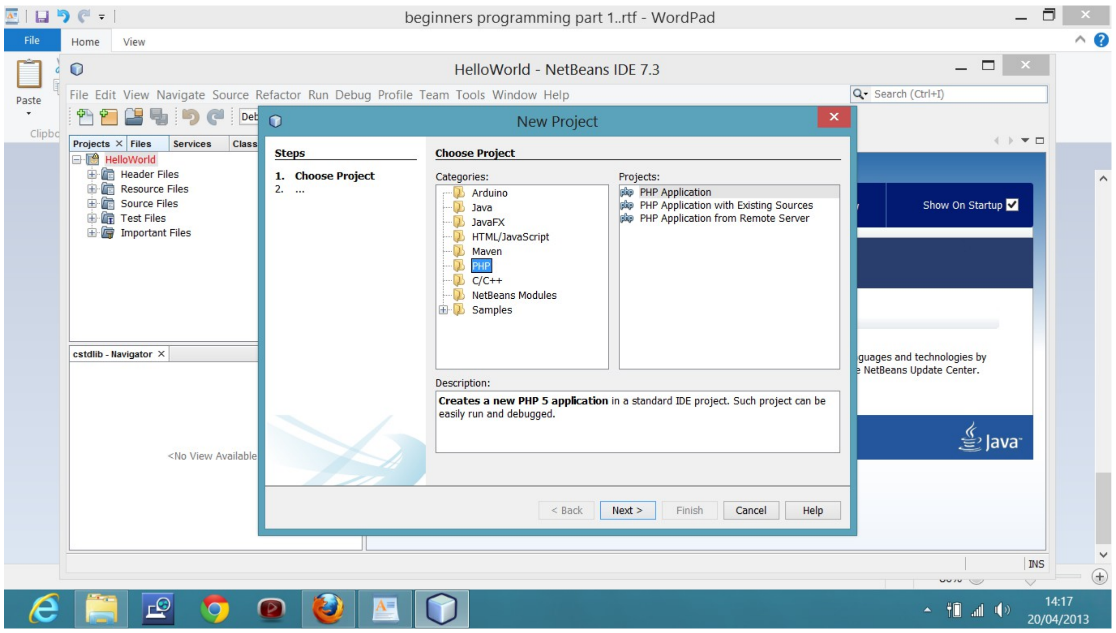

beginners programming part 1..rtf - WordPad
HelloWorld - NetBeans IDE 7.3
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
HelloWorld
Header Files
Resource Files
Source Files
Test Files
Important Files
New C/C++ Application
Steps
1. Choose Project
2. Project Name and Location
Browse...
< Back
Next >
Finish
Cancel
Help

Project Name and Location

Project Name:	helloworld
Project Location:	C:\Users\dave\Documents\NetBeansProjects
Project Folder:	dave\Documents\NetBeansProjects\helloworld
Project Makefile Name:	helloworld-Makefile.mk
✓ Create Main File	main C++
Build Host:	
Tool Collection:	Default (None (No Compilers Found))

Project Folder already exists and is not empty. "nbproject" exists.

for those writing in PHP you will obviously select PHP application.


New Project

Steps
1. Choose Project
2. ...

Choose Project

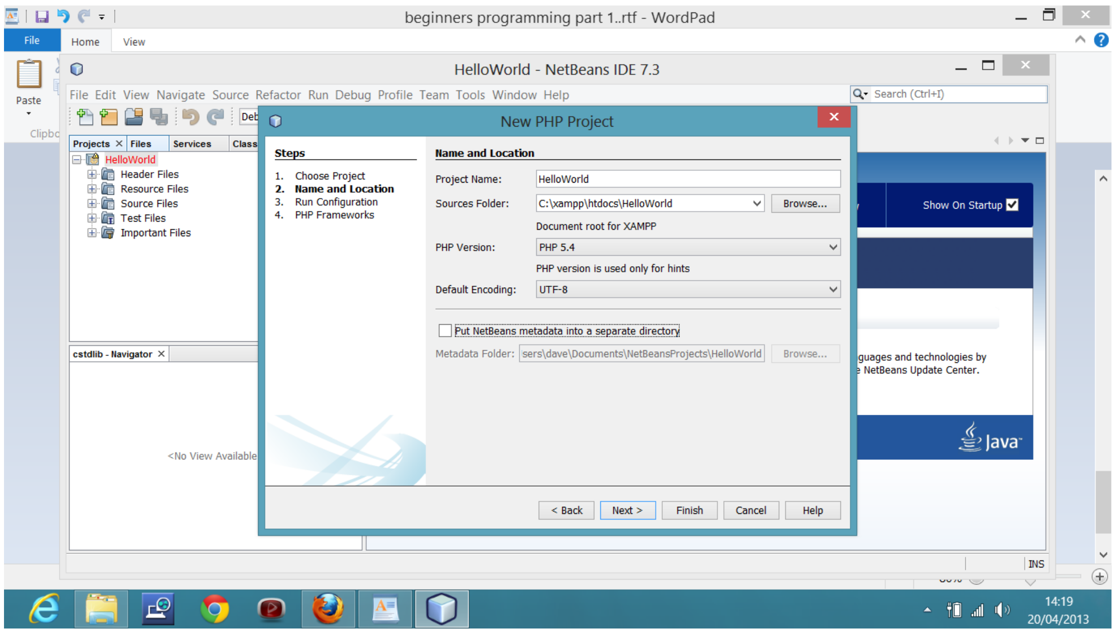
Categories:
- Arduino
- Java
- JavaFX
- HTML/JavaScript
- Maven
- PHP
- C/C++
- NetBeans Modules
- Samples

Projects:
- PHP Application
- PHP Application with Existing Sources
- PHP Application from Remote Server

Description:
Creates a new PHP 5 application in a standard IDE project. Such project can be easily run and debugged.

beginners programming part 1..rtf - WordPad
HelloWorld - NetBeans IDE 7.3
< Back
Next >
Finish
Cancel
Help

And guess what you will also be prompted to name your project. So again name it Hello World.


New PHP Project

Steps
1. Choose Project
2. **Name and Location**
3. Run Configuration
4. PHP Frameworks

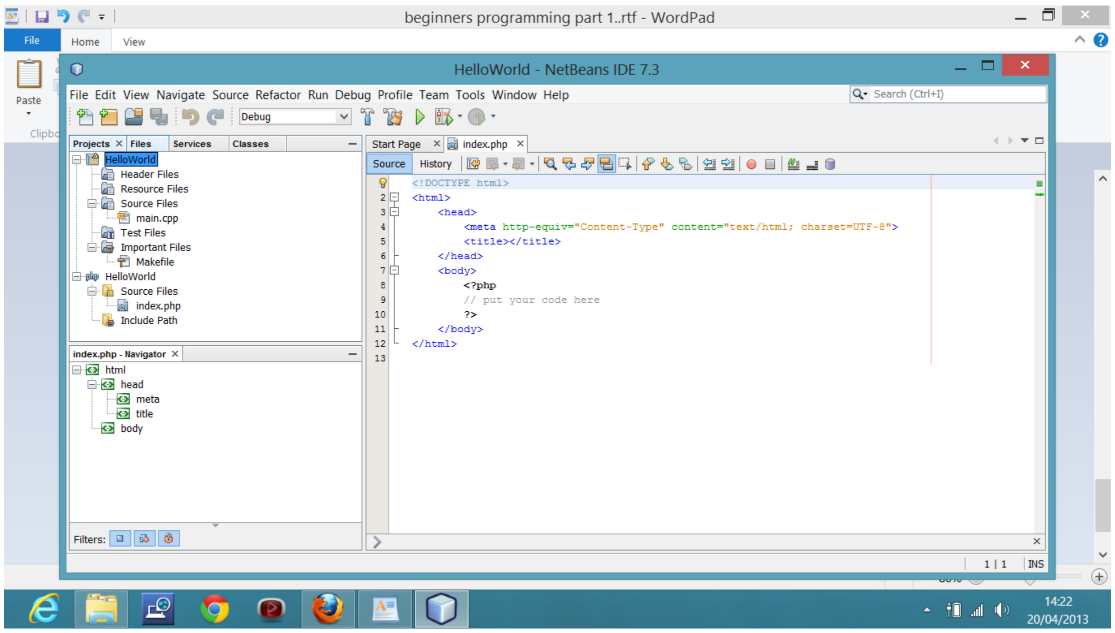
Name and Location

Project Name: HelloWorld
Sources Folder: C:\xampp\htdocs\HelloWorld
Document root for XAMPP
PHP Version: PHP 5.4
PHP version is used only for hints
Default Encoding: UTF-8
Put NetBeans metadata into a separate directory
Metadata Folder: sers\dave\Documents\NetBeansProjects\HelloWorld

beginners programming part 1..rtf - WordPad
HelloWorld - NetBeans IDE 7.3
Browse...
< Back
Next >
Finish
Cancel
Help

Right now we are ready to write out first program.

Now you may have noticed that you now have a in the left hand pane of your IDE there is what appears to be a directory tree. This is your basic stucture for your project.



```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title></title>
    </head>
    <body>
        <?php
        // put your code here
        ?>
    </body>
</html>
```

In the C++ project you will notice that a host of directories have been created we can ignore them for the time being we are only inteseted in the file main.cpp which has been created in the Sources folder.

In the PHP project you will notice that it has created a index.php file again under the source files folder.

So lets open the these files, and see what the IDE has created for us.

Main.cpp	Index.php

*
 	
* File: main.cpp
 	
* Author: dave
 	
*
 	
* Created on 20 April 2013, 14:14
 	
*/
 	
#include <cstdlib>
 	
using namespace std;
 	
/*
 	
*
 	
*/
	<!DOCTYPE html>
	<html>
	    <head>
	        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
	        <title></title>
	    </head>
	    <body>
	        <?php
	        // put your code here
	        ?>
	    </body>
	</html>

```
int main(int argc, char** argv) {



   return 0;

}
```

Now the first thing you may notice is that different lines are appear in different colours now you may ask what that is about. It is to aid you in identify different sections and functions   of your code.

Comments

 If we look at the c++ sample we can see that it starts with a comment. In C++ comments are represented by a line starting a double back slash // or by a section starting with a /* and ending with a */ as you will see in the IDE these are represented as grey text (you can edit these colours but I would always recommend staying with the default colours) .

Now if we look at the PHP sample we can also see a comment line again in grey where it is telling you to insert you code hear as you can see this also starts with a double backslash // and again is grey so we can now see that comments in the IDE are grey in colour.

Include directive

Now the next line of the c++ program contains an #include line saying #include <cstdlib> and this is in green. The IDE is recognising this as an an **include directive**. It tells the compiler and the linker that the program will need to be linked to a library of routines. In this case the c standard library <cstdlib>. You will learn much more about libraries of code later in this course.

Using statements

This statement is called a *using* directive. The latest versions of the C++ standard divide names (e.g. cin and cout) into subcollections of names called *namespaces*. This particular *using* directive says the program will be using names that have a meaning defined for them in the std namespace (in this case the iostream header defines meanings for cout and cin in the std namespace).

Some C++ compilers do not yet support namespaces. In this case you can use the older form of the include directive (that does not require a *using* directive, and places all names in a single global namespace):

```
#include <iostream.h>
```

You may now have noticed that this is in **Blue** as are some other parts of the c++ text. This is because they are **reserved word** in the programming language and have specific meaning and you cannot use them to name your variable , object, classes etc.(There will be more on this later)

If we switch back to the PHP sample file you will notice that it has also marked all of the HTML tags in Blue as well as they are reserved terms with specific meanings.

First Program

So now lets look at actually adding some code to these empty project files to create our hello world sample.

Main.cpp	Index.php
In this file add an new include statement under the one which already exist. #include <iostream> You have now added the the input output library to your project. Now you will need to add code to write the output "hello world" to do this you will need to go the the section of the program between the start section marked by an open curly bracket { but before the return statement. In here type count<< " Hello World"	In the PHP example add the following below the comment //put your code here echo " Hello World" Make sure you have started you Apache and Tomcat servers using xampp

Now in your IDE make sure you have you have the file you are working on open. Now if you look at the menu you will notice a RUN menu and a DEBUG menu. (We will go into this in more detail later) you will also notice a green arrow icon on the menu bar below. Click on this green arrow and you will have run your first program.

Those writing in C++ will notice that a new window appear at the bottom of the IDE. This is where your output will appear. And you will see

```
Hello World
RUN SUCCESSFUL (total time: 94ms)
```

Those writing in PHP a browser should open with you a web page which says guess what 塗ello world�

Now we have written our first program you may ask yourself why did I bother going to all that effort when I could have just type 滴ello World� And this is the main issue with the 滴ello World�example it does not teach you anything useful.

So lets look at something a bit more useful.

Variables and simple Data Types

Variable are used to store different kinds of data (or values). These values can be numbers, text, or much more complex data. The table below will show you the data types available in each language.

PHP	meaning	C++	
Boolean	Logical TRUE or FALSE	`bool`	Boolean value. It can take one of two values: true or false. Size: 1byte values: true or false
		`char`	Character or small integer. Size;1byte values: signed: -128 to 127 unsigned: 0 to 255
		`wchar_t`	Wide character. Size:2 *or* 4 bytes value: 1 wide character
Integer	Whole numbers (e.g., 7, 78, –132, 87348)	`short int` (`short`)	Short Integer. Size:2bytes values : signed: -32768 to 32767 unsigned: 0 to 65535
		`int`	Integer. Size: 4bytes

			values: signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
		`long int` (`long`)	Long integer. Size:4bytes values: signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
Float (double)	Numbers with decimal notations (e.g., 78.23, -3.25 or 348.125)	`float`	Floating point number. Size:4bytes values: +/- 3.4e +/- 38 (~7 digits)
		`double`	Double precision floating point number. Size:8bytes values: +/- 1.7e +/- 308 (~15 digits)
		`long double`	Long double precision floating point number. Size:8bytes values: +/- 1.7e +/- 308 (~15 digits)
String	Characters, letters, or numbers, defined within double or single quotes (e.g., "Hy there" or '123AvR')	String	Characters, letters, or numbers, defined within double or single quotes (e.g., "Hy there" or '123AvR')
Array	A variable that can hold multiple, separate pieces of values. It's like a list of values, each value being a string or a		

	number or even another array.		
Object	The instance of a PHP class. The basics for class definitions and object-oriented programming.		
Resource	Stores a reference to functions, databases, files, or other resources outside of PHP		
NULL	Defines a variable with no value; the variable exists, but contains nothing (not an empty string, not the value 0, nothing)		

In C++

Declaration of variables

In order to use a variable in C++, we must first declare it specifying which data type we want it to be. The syntax to declare a new variable is to write the specifier of the desired data type (like int, bool, float...) followed by a valid variable identifier. For example:

```
int a;
float mynumber;
```

These are two valid declarations of variables. The first one declares a variable of type *int* with the identifier *a*. The second one declares a variable of type *float* with the identifier *mynumber*. Once declared, the variables *a* and *mynumber* can be used within the rest of their scope in the program.

If you are going to declare more than one variable of the same type, you can declare all of them in a single statement by separating their identifiers with commas. For example:

```
  int a, b, c;
```

This declares three variables (*a*, *b* and *c*), all of them of type *int*, and has exactly the same meaning as:

```
int a;
int b;
int c;
```

The integer data types *char*, *short*, *long* and *int* can be either signed or unsigned depending on the range of numbers needed to be represented. Signed types can represent both positive and negative values, whereas unsigned types can only represent positive values (and zero). This can be specified by using either the specifier *signed* or the specifier *unsigned* before the type name. For example:

```
unsigned short int NumberOfSisters;
signed int MyAccountBalance;
```

By default, if we do not specify either *signed* or *unsigned* most compiler settings will assume the type to be signed, therefore instead of the second declaration above we could have written:

```
  int MyAccountBalance;
```

with exactly the same meaning (with or without the keyword `signed`)

An exception to this general rule is the *char* type, which exists by itself and is considered a different fundamental data type from *signed char* and *unsigned char*, thought to store characters. You should use either `signed` or `unsigned` if you intend to store numerical values in a *char*-sized variable.

`short` and `long` can be used alone as type specifiers. In this case, they refer to their respective integer fundamental types: `short` is equivalent to `short int` and `long` is equivalent to `long int`. The following two variable declarations are equivalent:

```
short Year;
short int Year;
```

Finally, `signed` and `unsigned` may also be used as standalone type specifiers, meaning the same as `signed int` and `unsigned int` respectively. The following two declarations are equivalent:

```
unsigned NextYear;
unsigned int NextYear;
```

PHP

Using variables

Variables are used for storing values, data like text strings, numbers or arrays.
When a variable is declared, it can be used over and over again in your script.
Regardless of what type you are creating, all variables in PHP follow certain syntactical rules:

- Variable names have to begin with a dollar sign ($). For example, $name
- After the dollar sign ($), the next character in the name must be a letter or an underscore; after this, the remainder of the variable name can be any combination of letters, numbers, and underscores. For example, $var_name1
- The name of a variable is case-sensitive, so $varname is a different variable than $varName
- The variable named $this is reserved for use in Object Oriented PHP, so it can't be used elsewhere

Variables can be assigned values using the equals sign (=).

```
<?php
$var_name = value;
?>
```

There are two ways to assign values to variables: by value and by reference.

Assign by value

The typical way to do assignments is to define by value. Value can be a number, a string (or any data type), or another variable previously defined.

```
<?php
$var1 = 8;                          // the value is a number
$var2 = 'coursesweb.net';       // the value is a string
```

```
$var3 = true;                          // a boolean data
$var4 = $var1;                         // the value of $var4 is 8 (the
value of $var1)

echo $var4;                          // 8
?>
```

- The values of these variables remains intact until it is reassigned or the script has completed.

• If you assign another value to an existing variable, the new value will overwrite the old one.

```
<?php
$site = 'coursesweb.net';
$site = 789;                           // assigns another value, a number

echo $site;                           // 789
?>
```

• Variables can be printed within double quotation marks, but not within simple quotation marks. If you add a variable inside simple quotation marks, will display its name (not its value).

```
<?php
$name = 'Marius';
echo "Hy $name". '<br />';           // Hy Marius
echo 'Hy $name';                     // Hy $name
?>
```

• Dot character (.) is used to join strings.
Example:

```
<?php
$protocol = 'http://';
$site = $protocol . 'coursesweb.net';
echo $site;                  // http://coursesweb.net
?>
```

Assign value by reference

The reference approach allows the content of a variable to affect the value of another variable (previously defined), or a function to affect a variable that is not part of that function (see the lesson: Passing variable to function by reference).
To define a variable by reference, simply add the ampersand (&) character before the dollar sign ($).

```
<?php
$var1 = "Tutorials";                    // assigned by value
$var2 = &$var1;                         // $var1 is assigned to $var2 by
reference
```

```
echo $var2;                // Tutorials

// changing the value of $var1 will modify also the $var2
$var1 = 'www.marplo.net';
echo '<br/>'. $var2;         // www.marplonet.net

// changing the value of $var2 will be transmitted to $var1 too
$var2 = 'http://coursesweb.net';
echo '<br/>'. $var1;                    // http://coursesweb.net
( because of the "by reference" )
?>
```

- Once a variable is assigned by reference, it is tied to its referenced variable. If one of them changes, will transfer the same change to the other one. If you change the value of $var1 will change the value of $var2 too.
Because assigning values by reference can complicate the script, it's better to avoid this.

Dynamic variables

A dynamic variable is named using two dollar signs ($$) and is associated with a normal variable that has a similar name.
The value of an usual variable gives the name (without the dollar symbol) to a second variable.

```
<?php
$good_people = 12;
$var_name = "good_people";
echo $$var_name;                  // 12

// assigning a value to $$var_name will change the value to $good_people
$$var_name = 'other value';
echo '<br />'. $good_people;              // other value
?>
```

A dynamic variable does not contain its own value. Instead, it contains the location where you can find the value, in other words, the name of another variable.
The expresions: $var_name = "good_people"; and $$var_name; make result a variable "$good_people", the value of "$$var_name" is the value of "$good_people".

- Dynamic variables, as those by reference can be confusing and it's better to avoid them.

Review of Variable and Types

Now you will have notice that one on the main differences between PHP and C++ is that in C++ you need to identify the type of a variable where as in PHP you don't it is these idiosyncraticise that can confuse the people when they are beginning programming. All you need to remember is which one use when and if you get it wrong the IDE will show you you have a mistake and highlight it.

To prove this case let do a quick edit to our hello world sample program we wrote earlier.

Main.cpp	Index.php

In the main body of program insert the following
 	
`cout << "Hello World"<< endl;`
`string MyHello ="Hello";`
`string MyName= "Dave";`
`cout<< MyHello +" "+ MyName;`
	In the main body of your program now update it to read
	
	`echo "Hello World" . '<br>';`
	`$MyHello="Hello";`
	`$MyName="Dave";`
	`echo "$MyHello $MyName";`

So now we have created a variable and then used that variable in printing an output. But again not really a very useful example.

You will also notice that the IDE colour codes the types in C# so you know that it is a type as turquoise, And for the observant among you may have noticed that the IDE also can offer you hints, prompts or suggestions based on what you are typing(this auto complete feature can be turned off but I would suggest leaving it on.

Dice Game

So we how do we actually go about creating a program which is actually useful.

So to that end lets start looking at an actually real world problem. How about a simple dice game.

We will start by identifying what makes a Dice a Dice. How could you describe a dice, well is a cube, it has 6 faces or does it. If you have ever played a War game or a Role Playing Game you may very well have encounter dice which have a different shape.

A D4 for example has 4 faces and is a triangular pyramid . So perhaps all we can say is a dice has a shape and a number a flat surface faces. But a dice is also physical thing it will also have a mass and a weight. Or in other word it is a **OBJECT** and this is where we get the term Object-oriented Programming. So how would we define our dice. I say why make things complicated let start off with only defining a dice by it faces and its shape.

So how would we do that you might ask we this is where we meet the Class. People can quite often use the terms object and class interchangeably this is not actually correct, a Class is a description of an Object if you wish and an Object is an instance of a Class. So how would we actually   do this

Dice Class

Main.ccp	Index.php
class dice	class Dice
{	{
public: int faces;	public $Faces;
public: string shape;	public $Shape;
}	}

So we now have a class defined but it does not actually do anything. So what may we want to do.(split into pairs and see what things you may want to be able to do with a dice.

Ok so we now have a list things we want to be able to do or   functions. As they are known.

How would we add a function to a Class.

Main.ccp	Index.php
class dice	class Dice
{	{
int faces;	public $Faces;
string shape;	public $Shape;

```
public:void set_values(int  ,string);
public: int diceThrow();


};

void CDice::set_values (int a, string b) {
 faces = a;
 shape = b;
};
```

```
 public function
_construct($numberOfFaces,
$shapeOfDice)
    {
       $this->Faces=$numberOfFaces;
       $this->Shape=$shapeOfDice;
    }

 public function throwDice()
   {
      $this->thrownDice=rand(1,$this-
>Faces);
   }
}
```

Now you create the rest of the functions you feel you require. (perhaps to be able to display the shape of the dice and display the number of side and also to display and throw the dice.

Now in order to throw the dice you will need to know about one of the built in functions of both programming languages and that is the RAND function. I don't really want to get bogged down with the theory of random numbers and how no number is truly random as any mathematician will tell you . So let just take it as read, for our purposes here that someone else has already developed and sorted out a way for us to generate a random number. And that way is to use the RAND function.

c++	php
Rand() % 100 ; will for example select a random number between 0 and 100. if you want to select between the different values Rand()%100+4 will for example pick a number between 4 and 100. so for a six sided dice throw you would use Rand()%6+1	rand(initial value, final value) will pick a number between the initial value and the final value. So for a six side dice you would use rand(1,6);

So that will conclude this lesson.

What have we learned

We have learned how to create a new project.

We have learned how to produce an output to the screen

We   have cover the   basic data types available in both languages.

We have created variable and used to hold a data type and then output the value to the screen.

We have created a class

We have created a function

We have used the class and outputted the values to the screen

We have used the function and outputted the values tot he screen
