Week 2 Module 4 Data Types and Expressions Part 2

Float and Double Data Types 1.2

Ok so we talked in quite a detail about the int data type. Let's start talking about float and double. These two types are quite similar to one another so we will talk about both of them together. And we will just say the small differences between them are. So float and doubles are used both to store real numbers in them. Real numbers meaning number that can have fractional parts. Let's say 7.5 or 3.65 some decimal point in them. Let's try and think about the inner representation in of a data that can have fractional part. So as integers both float and doubles also have a fixed size it is not that their size changes depending on the number they are represented. Each float has a size of its own and each double has a size of its own. They differ from one another with their sizes. So for doubles each double data takes 8 bytes where each float data takes only 4 bytes. So double is double the size of a float but again each double data takes 8 bytes no matter what's represented in it and each float data takes 4 bytes no matter what represented in it.

Float and Double Data Types 1.3

For example if we have an integer x and a double y ints takes 4 bytes so x takes let's say from 100 to 104. And double y y is a double so y takes 8 bytes from 104 to 112. If we assign x with 6 and y with let's say 7.658 so we are thinking that x would contain 6 and y would contain 7.658. But then we know that it is not we really write the number 6 and the number 7.658 in the memory. So how can we represent floating point numbers or real numbers with fractional parts using zeroes and ones.

Float and Double Data Types 1.4

One way of trying to do that is decide that if we have let's 8 bytes that the first 4 bytes would be for the integer part and the other four bytes would be for the fractional part. And this is what we call like where we put decimal point in a fixed position. After four bytes we have fixed position for the point. This is not how doubles and float are really represented inside the memory. I am not going to get into the details of the bits of how to represent floating point numbers you can read in the following link some more information about the floating points representation method it is called IEEE 754. But then I would say that the decimal point it is not in a fixed position the point can float around inside a bunch of bits that's why it is called the floating point representation method.

Float and Double Data Types 1.5

So just for an example if we have x equals 6 and y equals 7.658 so we know that 6 is represented in a 32 bit 2's complement because it is an integer and it would look like that. Y or 7.658 would be represented using the floating point representation method in case of a double in a 64 bit and it would look something like that. And therefore it would be stored in y just like that. So this is the way doubles and floats are basically represented using zeroes and ones. Let's talk about the built in datas that are considered to be doubles and floats. C++ literals of doubles and floats. So for doubles we just write 3.4 -8.975 or whatever decimal number we want to write. If we want to write an integer number but to be considered as double we have to write 6.0 to add a .0 to it if we just write 6 the compiler would treat it as an integer. If we want to write the double 6 as a C++ literal we just write it 6.0. So for doubles we basically write the decimal representation of real numbers. For floats we just add a prefix of f at the end of the numbers so the compiler would know that when we write 3.4f it is the float literal with a value of 3.4 and not the double literal of 3.4. That's also for -8.975 and so on. So that's how we create C++ literals regarding arithmetic operators in this case very very expected. We have a plus minus multiplication division. In this case the slash operator would do real division it won't do div as it had in

integers. When we place divide operator between two doubles or two floats it would make the real division of these two values and assignment as we have talked before.

Float and Double Data Types 1.6

Ok let's use these types in order to implement the following program. Let's write a program that reads from a user the radius of a circle and then the program would calculate and print the area of this circle. So for example we ask the user to enter radius the user would say I don't know 2.6 and then the program would respond the area of a circle with radius of 2.6 is in this case 21.2372. Let's take a minute and think how we would implement it. So kind of straight forward we can just use the formula to calculate the area of a circle. So if we have a circle with a radius of r the area of the circle is just pi times r squared. Let's use this formula in order to make our calculation. Let's go ahead and implement it.

Float and Double Data Types 1.7

Ok let's go ahead and implement this program. So first thing we do is we ask the user to enter the radius of the circle. So cout please enter the radius so break the line here. And then let's read the user's radius for that let's first declare a variable. Since radius could be a real number with a fractional part let's use the double type for this variable. And if we are thinking how to name it this is going to store the radius maybe we should just name it radius. So double radius that's a variable we created here and let's read into the radius variable. So we got the input from the user now we should calculate the area of this circle. So let's add a variable to store the area and I will just name it area. And then after having the radius let's set area to be pi times radius squared. So pi is 3.14 times radius times radius. Something like that so pi that's radius squared. So we have the area calculated after that we should announce to the user that the area of a circle is whatever the value of the variable area is. So let's start cout and then let's print the text the area of a circle with radius of space here. And then write the value of the variable radius and then the word is once again space before and after and the value of the variable area let's end the line. So we have the area of a circle with the radius of the value of the variable radius is the value of the variable area. Basically output statement we want to make here let's test it see if we didn't make any mistakes. Please enter the radius 2.6 and then it says the area of a circle with radius of 2.6 is 21.2264. One thing I would change here is instead of hardcoding the value 3.14 let's create a constant with this value and give it a meaningful name. In this case we probably should name it pi. Because 3.14 stands for pi so up here above the main we will create a constant of type double named pi and the value would be 3.14. After we have that instead of writing 3.14 we should just use pi the constant name and that would be replaced by 3.14 and that makes our code be more readable. So we say that area equals pi times radius squared. Let's test it once again although it is not supposed to behave any differently. So the radius is 2.6 and then it prints that the area of a circle radius is 2.6 is once again 21.2264. So looks very good though that is not really an accurate value here for the area that's because the value of pi I gave here 3.14 is not very good estimation for pi. So we could either have pi figure out what the more accurate value of pi and just change our constant. Or we could add or use some predefined value of pi. So pi is not a pre defined constant in C++ but then there are extensions for C++ language that we can use these extensions include the value of pi. In order to use these extensions we use the pound include statement just as we've done up here pound include iostream. Baiscally we are extending the C++ language to include some more abilities of io input output. Actually cout and cin are not built in operators in C++ they are not defined not predefined in C++. If we won't have this include iostream cout and cin won't work we will have compilation errors for trying to execute them. In order to use cout and cin we need to extend our language to include some iostream capabilities and then cout and cin are defined then we can definitely use them. Same thing with pi in order to have the predefined constant pi and to be able to use it we need to include to extend our language. So let's add some include statement

here and the name of the library that we are going to use the pi constant and some other mathematical capabilities is called cmath. So if we include cmath that would have predefined pi constant in it so we won't need to define this constant on our own. We can just use the cmath pi constant that is already defined. The only thing is that it is not called pi p i as we could expect this is called m underscore pi math pi. So area is now math pi times radius times radius. This math pi constant is much more accurate than 3.14 and if we execute it again the program will now we are expecting to get a better approximation for the area it would be slightly different than 21.2264. Let's try it out if we have 2.6 the area now is 21.2372 slightly different probably better result here. Yes so we basically read the radius from the user we use the pi constant in order to calculate the area and then we output the statement saying what the area is.

Type Casting 2.1

Ok now that we have a few types in hand int float double let's see what happens in some cases when we try to mix these types kind of together. So for example we have four variable two if type int x one and x two and two of type double y one and y two. If we assign x one with 6 and y one with 6.7 that would a perfectly valid because six is an int literal x one is an int variable and assigning int into an int makes perfect sense. Same thing when we are assigning y one is 6.7 6.7 is a double literal y one is a double variable and the assigning a double into a double is very reasonable. But what would happen if we try to assign y two with a value of 6? When we try to assign an integer 6 an integer literal into a double y two which is a double variable. Formally this expression is legal we cannot assign an int to a double or something that is not of the same type. The compiler won't say anything it will figure out a way to get over that but formally what we need to do is basically to convert the representation of the integer 6 to the representation of the same or the equivalent value in the representation of a double. The syntax to do that is called casting converting the representation of data to one type to an equivalent value in another type. The syntax is the new type included in parentheses in this case double followed by the value we want to convert into double. So the expression double 6 would convert the integer 6 into a representation or an equivalent representation in a type double. And this expression double 6 is of type double and then assigning double 6 into y two is a legal assigning. It is a assigning a double into a double variable. So when a casting is evaluated basically the representation of this data is converted from one representation method to another. From 32 bit 2's complement representation of an integer in this case into a 64 floating point representation of a double. And this conversion is done by syntax of casting into a double of the value 6. It would be interesting to think what would happen if we tried to convert or to cast 6.7 the double 6.7 into representation of int. Int of 6.7 but then we can't really keep the value or the converted value equivalent to 6.7 because integers cannot represent a fractional part or a number a real number 6.7. So what would the compiler convert the double 6.7 to? To what integer would it be converted? So we can try it out but I will tell you what is going to happen. It is basically going to remove the fractional part and convert 6.7 to a 6 in this case. So this assigning x two equals int of 6.7 would assign the integer x two to the value of the integer 6.

Expressions 2.2

Let's take a look at another example of what happens when we mix the types. For example if we cout 5 divided by 2 so since 5 and 2 are both integers the dividing operators has the meaning of div. As we said that when slash comes between two integers that means div. If we do 5.0 divided 2.0 the dividing operator has the meaning of real division because when a dividing operator comes between two doubles it has the meaning of a real division. So the first cout would print 2 because 5 div 2 is 2. The second cout would print 2.5 because 5.0 divided by 2.0 is 2.5. What do you think would happen if we divide 5.0 by 2? When we try to divide a double by an integer. What meaning would the divider

operator have in this case? Actually I don't know maybe it has the meaning of the div maybe it has the meaning of a real division or maybe it could even be an error. So it is not an error you can relax here the compiler when we have mixed types in an expression the compiler would try to resolve the issue of the mixing types and would try to make casting of the arguments to state where the two operands would be of the same type. The casting would be what we call implicit cast casting that do not lose any accuracy in this case converting an int to a double never loses any accuracy. Because every integer could be represented as a double not the other way around right not every double can be represented as an integer some accuracy can be lost. Just as when we converted 6.7 into an integer the .7 got lost there. But converting 2 to a double that is a conversion that can be done implicitly can be done without losing any accuracy. So the compiler would try to match the types of these operands 5.0 and 2 in this case casts 2 to a double and then the dividing operator would be in the meaning of real division. So this cout would also print 2.5. Another thing I want to show you here what happens if we assign x to the value of 5 divided by 2. Obviously no issue 5 and 2 are both integers the dividing operator is div in this case so 5 div 2 is 2 and x as an integer gets the value of 2. But what will happen if we assign y to 5 div 2? Y is a double right so do you think that in this case 5 divided by 2 would be 2.5 and y would get the value 2.5 or 5 div 2 5 divided by 2 would still be 2 and y would be assigned to 2. You can try it out but I will tell you what would happen. So an assignment expression has two steps in its evaluation. First step is evaluating the right hand side after we get this value the value is assigned to the variable. And the first step doesn't take into account what's the type of the variable we are going to assign the second step. The first step just evaluates the right hand side in this case evaluating 5 divided by 2. And once again since 5 and 2 are both integers the dividing comes in a form of div. So when we are evaluating the right hand side here 5 divided by 2 would result to 2 the integer 2. But then we are trying to assign the integer 2 to the variable of type double formally we should have casted the integer 2 to a double but the compiler would do it for us so y would get the double of the value 2. What we would like to write as the literal 2.0. So y would be the double 2 in this case.