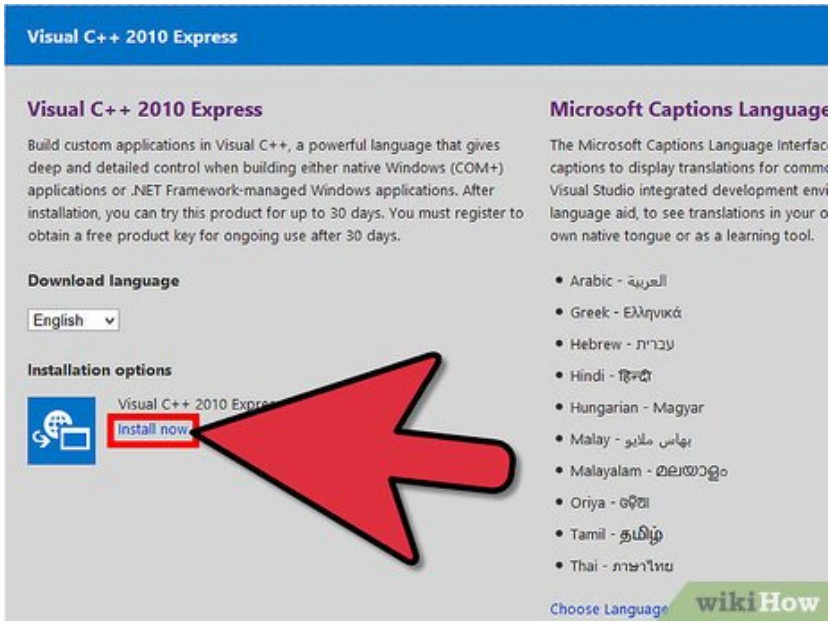
How to Calculate Your Wage in C++

You may find it useful or interesting to know how much you will earn in a month or year. While it is possible to do this calculation manually or with a calculator, writing a program is useful to understand what you're doing and to repeat...

Part 1 of 2:

Getting a Compiler

1. 

   Install Microsoft Visual Studio on Windows. It is an IDE that can be downloaded free of charge from the Microsoft website. Any version will work for this program, for example Visual Studio Express 2013.
   1. Note that you must have or create a Microsoft account to download and install Visual Studio. If you don't want to or can't, choose another program.
2. **Use GCC on Linux or to avoid creating a Microsoft account.** See the article on how to compile a C program using the GNU Compiler for more detailed instruction for both Linux and Windows.
   1. GCC is pre-installed on most Linux distributions. If not, try installing it from your package manager or building from source.
   2. On Windows, you have to install MinGW to get this compiler. It's more difficult to set up and use, especially if you're not familiar with the Windows command line, but it doesn't require a Microsoft account and is open source.
3. **Install Xcode on Mac OS X.** XCode is an IDE provided by Apple, and it includes the Clang compiler for compiling C++ code.

4. **Use another compiler or IDE if you prefer.** The above are only suggestions, if you prefer using some other technology, this code will also work.

Part 2 of 2:

Writing the Code

1. **Start a new project or open a new file.** The exact way to do this is different in every IDE, but usually you select something like "New Project" in a menu, enter some information and a new file opens. If you have only a compiler and no IDE, open the file in any text editor.
2. **Explain what the code does with a comment.** To make a comment, put // in front of a line. If you're using an IDE or text editor with syntax highlighting, the line will change its colour. Comments explain what the code below or next to them does. The program just ignores them, and they are not necessary for it to work, but it's good practice to use them so that you or other people can understand the code better. Write something like this as the first line of your code:

   ```
   // Wage program. Takes hourly wage and calculates monthly and yearly wage.
   ```

3. **Include the IOStream library.** IOStream is the library that allows a C++ program to accept inputs from the keyboard and output them to the screen. Write the following below the comment:

   ```
   #include using namespace std;
   ```

4. **Add the main function.** When you run the program, it does everything written in the main function. Note that the main function *must* be named `main`, otherwise the program will ignore it. Declare the main function as `int`. While the main function could have any other data type, it's most common for it to be `int`. The curly brackets indicate the beginning and the end of the function. To make the code more readable, you may indent everything inside the brackets with spaces, but that's not required.

   ```
   int main() { }
   ```

5. **Create variables.** Variables are where information is stored. For this program, you need to create variables that store the information the user enters and variables that store the results of the calculations the program does. Declare all variables as "float" by writing `float` in front of them. This will tell the program that you'll store numbers with a decimal point in them, like 3.7 . The program will still work if the user enters a number without a decimal point, it just stores that number with a decimal point internally.

   ```
   float wage; float hours; float overtime; float weeklywage; float
   yearlywage;
   ```

6. **Get the user's input.** To calculate the user's weekly and yearly income, the program needs to know their hourly wage and hours they worked.
   1. Explain what input you want with `cout`. The user should know what they have to type in when. For example, to ask for the user's hourly wage, you could write:

      ```
      cout  "Please enter your hourly wage:"  endl;
      ```

   2. Get what the user types and store it in a variable. This is done with `cin`. Note that you must have declared the variable somewhere else before using it for `cin`. For example, to store the hourly wage

the user types in in the variable `wage` you declared before, write:

```
cin >> wage;
```

3. Text must be between two double quotation marks (" "). This tells the program when the text starts and ends. The "endl" ends the line and produces a space between the output and input to help the user read the program better.

The complete code for this step looks like this:

```
cout  "Please enter your hourly wage:"  endl; cin >> wage; cout
"Please enter the hours you worked this week:"  endl; cin >> hours;
```

7. **Check whether the user worked overtime.** Do this with an if-else condition. It will assume that 40 hours is a normal work week. If the hours are above 40, it will calculate the overtime specially. If not, it will simply calculate with the normal wage. The if-else condition looks like this:

```
if (hours > 40) { } else { }
```

8. **Calculate the weekly wage.** This is done differently depending on whether there is overtime or not, but in each case, the final result is stored in the `weeklywage` variable.
    1. The program assumes that overtime gets 1.5 times more than normal wage. So calculate how much overtime there was, then add the normal work week hours multiplied with the hourly wage to the overtime multiplied with 1.5 times the hourly wage. The following code belongs between the curly brackets after the if-condition. The additional spaces for indentation are not necessary, but they make the code more readable.

    ```
     overtime = hours - 40; weeklywage = 40 * wage + overtime * 1.5 *
    wage;
    ```

    2. If there is no overtime, simply multiply the worked hours with the wage. The following code belongs between the curly brackets after the else-condition. Again, the additional spaces for indentation are not necessary, but they make the code more readable.

    ```
     weeklywage = hours * wage;
    ```

9. **Calculate the yearly income.** The program will assume that you work the same amount of hours each week, and that you have 50 weeks per year of either work or paid vacation. With these assumptions, multiply the weekly wage by 50 to get the yearly income.

```
yearlywage = weeklywage * 50;
```

10. **Output the results to the user.** If you followed the previous steps, the results are stored in some variables in the program. But the user doesn't see them yet. Again, you can use `cout` to print the results:

```
cout  "Your weekly wage is: $"  weeklywage  endl; cout
"Your yearly income is: $"  yearlywage  endl;
```

11. **Return 0.** The program would work without this, but it's common practice to return 0 if the program exits without any errors. This is also why the `main` function is usually declared as `int`. To return 0, simply write:

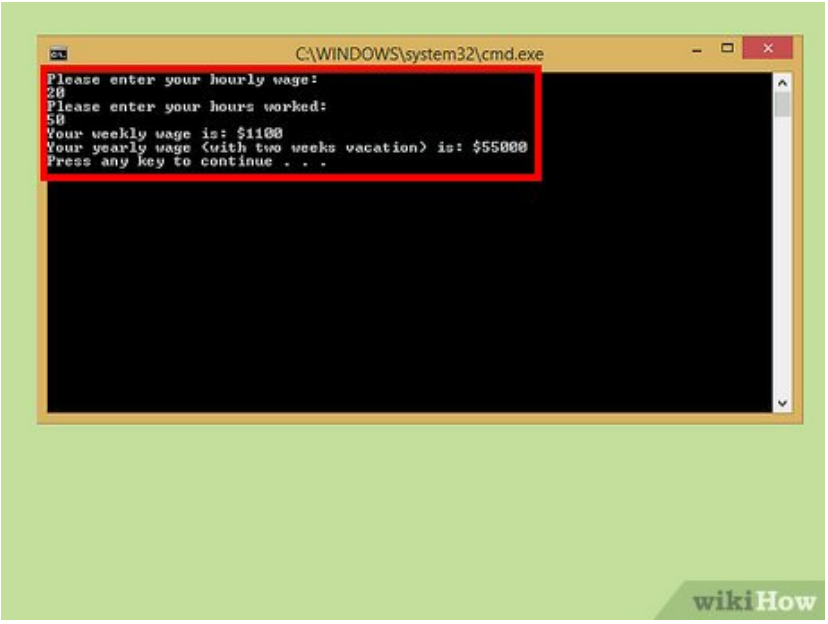
```
return 0;
```

12. **Look over your code.** While mistakes will be noticeable when you run the program, looking at the code you wrote again before running it can also help you spot them. The complete code should now look as follows:

```
// Wage program. Takes hourly wage and calculates monthly and yearly wage.
 #include using namespace std; int main() { float wage; float hours;
float overtime; float weeklywage; float yearlywage; cout
"Please enter your hourly wage:"  endl; cin >> wage; cout
"Please enter the hours you worked this week:"  endl; cin >> hours; if
(hours > 40) { overtime = hours - 40; weeklywage = 40 * wage + overtime
 * 1.5 * wage; } else { weeklywage = hours * wage; } yearlywage =
weeklywage * 50; cout  "Your weekly wage is: $"  weeklywage  endl; cout
  "Your yearly income is: $"  yearlywage  endl; return 0; }
```

13. 

Run the program. In Visual Studio, use the shortcut Ctrl + F5 to start the program without debugging it. Debugging is not necessary for this short and simple program. In Xcode, press ? Cmd + R. If you use GCC, save your file, type `g++ filename.cpp` into the command line (with `filename.cpp` being the name you saved your program under). On Linux, run the resulting file by typing `./a.out`; on Windows, open the resulting EXE file.

14. 

Calculate your weekly and yearly wage. This will let you test whether the program works correctly. When you run the program in Visual Studio, a black box (more formally, the command line) will come up. That is where your program runs. On any platform, it should now display the text asking for your wage and hours. You can type in any number. After getting your input, the program will go line by line. It will check whether the hours are more than 40 and then do the if-statement, and if not it will do the else-statement. However, all this will happen in the background and you won't see anything. It will then display the output to the user.