**C++ Introduction**

**Statement:** Type of instruction that causes the program to perform some action.

* End in ‘;’.
* Types of statement:
  + *Declaration*
  + *Jump*
  + *Expression*
  + *Compound*
  + *Selection (conditionals)*
  + *Iteration (Loops)*
  + *Try Blocks*

**Function:** Collection of statements that get executed sequentially (top to bottom).

* ***Identifier:*** Name of function.
* Note: Every C++ program must have a function named **main.** When the program is run, the **main** function is executed in sequential order.

**Cout: “**Character Output”

**Syntax:** Rules that govern how sentences are constructed in a language.

* **Syntax error:** Violating syntax rules of a language, detected by the compiler.

**Quiz:**

What is a statement?

*A statement is an instruction that performs an action.*

What is a function?

*A function is a collection of statements that gets executed sequentially.*

What is the name of the function that all programs must have?

*The* ***main*** *function.*

When a program is run, where does the execution start?

*The execution starts from top to bottom. X (*Sol. Execution starts with the first statement inside the **main** function.)

What symbols are statements in C++ often ended with?

*The ‘;’ or semicolon symbol is used to end statements.*

What is a syntax error?

*A syntax error is a compile error (*occurs at compilation) *that happens with incorrect grammar of a programming language.*

What is the C++ standard library?

*A library containing basic functions. (*Sol. A library file is a collection of precompiled code that has been “packaged up” for reuse in other programs. The C++ standard library is a library that ships with C++. It contains additional functionality to use in your programs.)

**Comment:** Programmer-readable note that is in the source code of the program. Used to help programmers document the code.

* *Single line comment:* Typed using ‘//’
  + Used to quick comments about **single** lines of code.
  + Example:
    - Std::cout <<”Hello World!\n”; // std:cout lives in the iostream library.
* Note: If lines are long, placing comments to the right can make your lines long, in that case single line comments are often placed ***above*** the line it is commenting.
* *Multi Line comments:* Typed using “/\* -- \*/”
  + **Beautify** example:
    - **/\*** This is a multi-line comment.  
       \* The matching asterisk to the left,  
       \* can make this easier to read.

\*/

**Warning:** Don’t use multi-line comments inside other multi-line comments. Wrapping single-line comments inside a multi-line comment is okay.

**Proper use of comments –**

Comments should be used for three things:

* For a given library, program, or function, comments should be used to describe ***what*** the library, program, or function, does. *Placed* at the top of the file orimmediately preceding the function.

*Example:*

A screenshot of a computer

Description automatically generated

**Second,** within a library, program, or function, comments can be used to describe ***how*** the code is going to accomplish its goal.

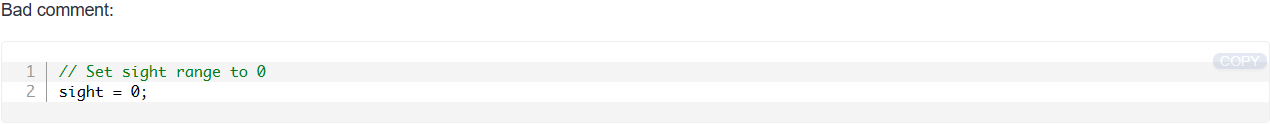
*Example:*

A screenshot of a computer

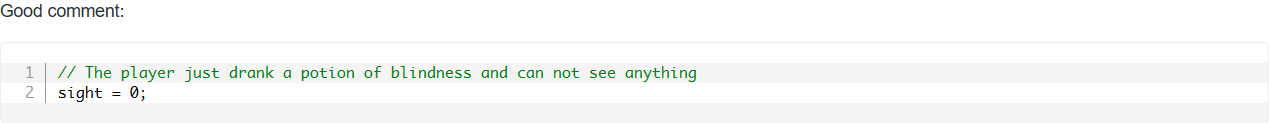
Description automatically generated

**Third,** comments can be used to describe ***why*** the code is doing something.

*Bad comments* explain what the code is doing. If you ever write code that is so complex that needs a comment to explain what a statement is doing, you probably need to *rewrite your statement,* not comment it.



*Reason:* We can see that the sight is being set to 0 by looking at the statement.



*Reason:* Now we know why the player’s sight is being set to 0.

*Comments are a good way* to remind yourself (or somebody else) why you chose to solve a problem one way instead of another.

A screenshot of a computer

Description automatically generated

**Commenting out code**: Converting one or more lines of code into a comment to temporary exclude parts of code from being included in your compiled program.

*Reasons to:*

* **Working on a new piece of code** that won’t compile *yet*, and you need to run the program. *Commenting out the code* *that won’t compile* will allow the program to compile so you can run it. When ready, you can uncomment the code, and continue working on it.
* **You’ve written some code** that *doesn’t work correctly*, and you don’t have time to fix it until later.
* **To find the source of an error**, it can sometimes to be useful to disable parts of your code to isolate the code that is *causing to program to not work* properly.
* **You want to replace one piece of code with another piece of code.** *Instead* of deleting you can comment your code until you’re sure your new code works properly.

For **Visual Studio Code**:

* *Comment a selection via* **Edit menu > Advanced > Comment Selection**

**Data:** Programs produce results by manipulating *data. Data* is any information that can be moved, processed, or stored by a computer.

**Value:** A single piece of data is called a *value,* e.g. a, 5 and text; Hello.

**RAM:** *Random Access Memory.* When a program is run it is loaded into the computer’s RAM. Common uses for this memory are to store values entered by the user, to store data read in from a file or network, or to store values calculated while the program is running (the sum of two values).

**Object:** An *object* is a region of storage (memory) that can store a value as opposed to direct memory access. A compiler would retrieve data stored in an object when a program is executed. An *object* with a name is called a ***variable.***

* **Note:** *Object refers to an unnamed object in memory, a variable, or a function. In C++ object has a narrower definition that* ***excludes functions.***

**Variable *instantiation*:** To create a variable, a declaration statement called a *definition* is used. Instantiation (Also called an *instance)* is creating an object and assigning it a memory address. Whenever the program uses an *instantiated* variable, it accesses the value of the variable in a specific memory location.

*Example:*



**Data Types**

* Integer (int): Any number written without a fraction component, such as 4, 27, 0, -2 are *integer variables.*

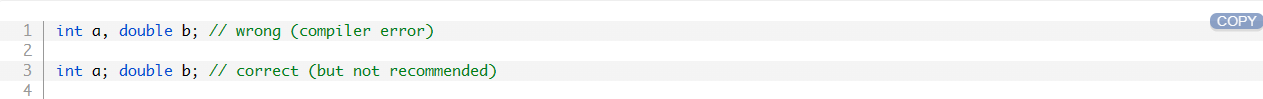
**Defining Multiple Variables:** It is possible to define multiple variables of the *same type* in a single statement separated by a comma.

*Example:*



*Incorrect syntax:*





**Quiz.**

* *What is data?*
* Data is any information that can be moved, processed, or stored by a computer.
* *What is a value?*
* A value is any single piece of data for example a number (5), letter (a) or text (hello).
* *What is a variable?*
* A variable is a named object. (Sol. Named region (*object)* of memory used to store values.
* *What is an identifier?*
* An identifier is the name of the object. X (Sol. An identifier is the name that a variable is assessed by.) In other words, the name of the variable.
* *What is a type?*
* A data type identifies what kind of data a variable is. (Sol. *A type tells the program how to interpret a value in memory.)*
* *What is an integer?*
* An integer is any non-fractional numeric value.