REMEMBER SEMICOLONS!!!!

Some Keywords:

Data Types:

`type identifier = initial_value;`

Note that you can assign the value with `(val);` *or* `{val};` *`instead of =`*

*Note declare more than one variable of the **same type**, using a comma-separated list:*

Or you can declare it unassigned: `type identifier;` (remember to assign it later)

You can also assign it using `auto;` (assigns it using the type of the `initial_value`)

- `int` - stores integers (whole numbers), without decimals, such as 123 or -123
- `float` - stores floating-point numbers, with decimals, such as 19.99 or -19.99 (4 bytes)
- `double` - stores floating-point numbers, with decimals, such as 19.99 or -19.99 (8 bytes)
- `char` - stores single characters, such as 'a' or 'B'. Char values are surrounded by single quotes
- `string` - stores text, such as "A". String values are surrounded by double quotes, are arrays
- `bool` - stores values with two states: true or false (T = 1, F = 0)

cin/cout:

cin gets a user input (`cin >> newvarname;`)

- Use `getline(cin, string)` to get the whole input, not just 1 word

cout outputs/prints text, DOES NOT ADD END-SPACE CHARS, use << to separate printed things (`cout << thing1 << thing2;`)

Comments:

Use // for single lines and `/* text */` for multi-line comments FINALLY

Constants:

`const type identifier = initial_value;`

CAN NOT BE CHANGED (will result in an error)

Strings:

Array access uses `''` (`txt[0] = 'T'`)

Arrays:

`type identifier[] = initial_value;`

You can also declare an array with a # in the [] and can add elements **up to that # - 1** (arr[#] = thing)

Starting Program:

```
#include <iostream>
using namespace std;  (can also be written before each thing → std::thing)
int main(){
      //Code here
      return 0; //To end the code
}
```

<u>Operators:</u>

Use % to get the remainder

You can use +=, -=, *=, /=

NOTE: for adding strings, you can use + or `.append()` but `.append()` is MUCH faster

<u>Built-Ins:</u>

- `txt.size() or txt.length()` – returns the length of the STRING (use `length()`)
- `max(thing1, thin2)/min(thing1, thing2)` – returns the max value
- `math` use `#include cmath`
  - `sqrt(#)`
  - `round(#)`
  - `log(#)`
  - `Any graphing calc function (tan, acos, floor, etc)`
- `If (condition) {\*Code here*/}` uses `else if {}` and `else {}`

```
switch(expression){
      case x:
            //code block
            Break;
      case y:
            //code block
            Break;
      default:
            //code block
}
```

<u>Loops:</u>

<u>While:</u>

`while (condition) {/* code here */}`

Can use `do {thing;} while (condition);`

<u>For:</u>

`for (statement 1; statement 2; statement 3) {/* code here */}` EXPLANATION

- `break` breaks the loop
- `continue` skips everything after it for that loop

<u>References:</u>

`string a = "Thing";`

`string &b = a;`

Now you can use both a and b to refer to a

NOTE: you can use &varname to get the memory address of that variable

Pointers:

`string a = "Thing"`

`string* b = &a`

Now b has the memory address of a (make sure the types match)

NOTE: you can use `string c = *b` to get the value at the memory address that b holds

Changing pointers:

Use `*pointername = "Thing"` to make the pointer the "Thing" location

Functions:

Declaring: `void myFunction(input_type input) {/* code here */}`

Calling: `myFunction();`

NOTE: can use `myFunction(input_type input = default_input) { }`

- Separate args with commas
- `return()` returns THE FUNCTION TYPE (***NOT VOID!!!***)

NOTE: you can pass through pointers to change the variables in the computer's memory

Function Overloading

Classes:

```
class MyClass {    // The class
  public:          // Access specifier
      //ACCESSORS
      int myNum;         // Attribute (int variable)
      string myString;  // Attribute (string variable)
      Const string& first() const {return var} //Access function

      //MODIFIERS
      void functionane(const string& varname){newvarname = varname};
};
```

Declaring: `myClass myObj  //Create an object of myClass`

```
// Access attributes and set values
  myObj.myNum = 15;
  myObj.myString = "Some text";
```

NOTE: you can declare functions inside of classes

Constructors:

A constructor is a special method that is automatically called when an object of a class is created

Use the class name followed by ()

Files:

```
#include fstream;

fstream filevarname(filename); //Open file

filevarname << "string" << endl; //Write to file

while (getline(filevarname, textvarname) { //Read from file

      vectorname.push_back(textvarname);

}

filevarname.close();
```

Copy Constructors:

assert()

Memory

Dynamic memory:

- Is created using the `new` keyword
- Accessed through pointers
- Removed through the `delete` keyword

ex) `int *p = new int; //Initiate`

```
*p = 17; //Assign value
cout << *p; //Print the value
delete p; //Delete the variable/memory space
```