#include <iostream>

using namespace std;

// main() is where program execution begins.

int main() {

cout << "Hello World"; // prints Hello World

return 0;

}a

**C++ Basic Syntax**

* Simple program which prints out whatever the input user programs into it. In this case, “Hello World” is what is printed out.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#include <iostream>

using namespace std;

main() {

cout << "Hello World"; // prints Hello World

return 0;

}

**Comments in C++**

* The difference between this program and the last is that the code will ignore **// prints Hello World** and the output will be “Hello World.”

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#include <iostream>

using namespace std;

int main() {

cout << "Size of char : " << sizeof(char) << endl;

cout << "Size of int : " << sizeof(int) << endl;

cout << "Size of short int : " << sizeof(short int) << endl;

cout << "Size of long int : " << sizeof(long int) << endl;

cout << "Size of float : " << sizeof(float) << endl;

cout << "Size of double : " << sizeof(double) << endl;

cout << "Size of wchar\_t : " << sizeof(wchar\_t) << endl;

return 0;

}

**C++ Data Types**

* The purpose of this code is to produce the correct size of various different data types on a person’s computer.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#include <iostream>

using namespace std;

// Variable declaration:

extern int a, b;

extern int c;

extern float f;

int main () {

// Variable definition:

int a, b;

int c;

float f;

// actual initialization

a = 10;

b = 20;

c = a + b;

cout << c << endl ;

f = 70.0/3.0;

cout << f << endl ;

return 0;

}

**C++ Variable Types**

* The purpose of this section is to learn the importance of the use of variable declaration which is to assure the compiler that there’s one variable which exists within a given type and name. This is needed so the compiler can compile without needing additional components and data about the variable. The purpose of this code is to show that even though the variable is declared at the top, it is later defined within the main function.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#include <iostream>

using namespace std;

int main () {

// Local variable declaration:

int a, b;

int c;

// actual initialization

a = 10;

b = 20;

c = a + b;

cout << c;

return 0;

}

#include <iostream>

using namespace std;

// Global variable declaration:

int g;

int main () {

// Local variable declaration:

int a, b;

// actual initialization

a = 10;

b = 20;

g = a + b;

cout << g;

return 0;

}

#include <iostream>

using namespace std;

// Global variable declaration:

int g = 20;

int main () {

// Local variable declaration:

int g = 10;

cout << g;

return 0;

}

**Variable Scope in C++**

* The purpose of this section is to learn the importance of scope variation which is a region which variables can be declared. This section goes through local variables which can only be used by statements that are within that function or segment of code.
* This section also discusses global variables which are defined outside of all functions. They are typically located at the beginning of the program and hold value all throughout program. (not just temporary)
  + The third segment of code is a program with the same name including global and local variables.

#include <iostream>

using namespace std;

int main() {

cout << "Hello\tWorld\n\n";

return 0;

}

#include <iostream>

using namespace std;

#define LENGTH 10

#define WIDTH 5

#define NEWLINE '\n'

int main() {

int area;

area = LENGTH \* WIDTH;

cout << area;

cout << NEWLINE;

return 0;

}

#include <iostream>

using namespace std;

int main() {

const int LENGTH = 10;

const int WIDTH = 5;

const char NEWLINE = '\n';

int area;

area = LENGTH \* WIDTH;

cout << area;

cout << NEWLINE;

return 0;

}

**C++ Constants/Literals**

* The purpose of this section is to learn the importance of constants and literals. It discusses the uses of integer literals, floating-point literals, Boolean literals, character literals. String literals, defining constants, #define preprocessors, and const.

#include <iostream>

using namespace std;

/\* This program shows the difference between

\* signed and unsigned integers.

\*/

int main() {

short int i; // a signed short integer

short unsigned int j; // an unsigned short integer

j = 50000;

i = j;

cout << i << " " << j;

return 0;

}

**C++ Modifier Types**

* The purpose of this section is to learn the importance and different types of data type modifiers including signed, unsigned, short, and long types. This program helps users identify the difference between how signed and unsigned integer modifiers are comprehended by C++.

#include <iostream>

// Function declaration

void func(void);

static int count = 10; /\* Global variable \*/

main() {

while(count--) {

func();

}

return 0;

}

// Function definition

void func( void ) {

static int i = 5; // local static variable

i++;

std::cout << "i is " << i ;

std::cout << " and count is " << count << std::endl;

}

**Storage Classes in C++**

* The purpose of this section is to learn the importance and different types storage classes which include register, auto, extern, mutable, and static. A storage class specifies the set life-time of variables and functions in C++.

#include <iostream>

using namespace std;

int main () {

for( ; ; ) {

printf("This loop will run forever.\n");

}

return 0;

}

**C++ Loop Types**

* The purpose of this section is to learn the importance and different types of C++ loop types which include while loops, for loops, nested loops, and do while loops. Loops are used for instances where a person may need to execute a segment of code a set number of times.

**C++ decision making statements**

* The next discussion the website goes through is decision making statements which describes if statements, if else statements, switch statements, nested if statements, and nested switch statements.

#include <iostream>

using namespace std;

// function declaration

int max(int num1, int num2);

int main () {

// local variable declaration:

int a = 100;

int b = 200;

int ret;

// calling a function to get max value.

ret = max(a, b);

cout << "Max value is : " << ret << endl;

return 0;

}

// function returning the max between two numbers

int max(int num1, int num2) {

// local variable declaration

int result;

if (num1 > num2)

result = num1;

else

result = num2;

return result;

}

#include <iostream>

using namespace std;

int sum(int a, int b = 20) {

int result;

result = a + b;

return (result);

}

int main () {

// local variable declaration:

int a = 100;

int b = 200;

int result;

// calling a function to add the values.

result = sum(a, b);

cout << "Total value is :" << result << endl;

// calling a function again as follows.

result = sum(a);

cout << "Total value is :" << result << endl;

return 0;

}

**C++ Functions**

* The final section discusses the different uses for functions which include function arguments, the default values for parameters, calling a function, and function declarations. Functions consist of a function header and body. Components include return types, function names, parameters, and function bodies. The programs above are examples of calling a function and giving parameters and values.