BASIC MAIN CLASS:

```
#include <iostream> //allows you to call cin/cout
#include <string> //allows you to use string
using namespace std;
int main()
{
   int num;
   //num will not be zero, garbage
   int z = 0; int k = 10;
   int x(9);
  cout << "Enter a " << “number: “ << endl;
  cin >> num;
  return 0; //NEED THIS STATEMENT
}
```

ESCAPE SEQUNCES:

\n = new line   \t = tab   \\ = “\”   \’ = “ ‘ “

READ USER INPUT:

```
#include <iostream> //allows you to call cin/cout
#include <string> //allows you to use string
#include <climit>
using namespace std;
int main()
{
   int num1; int num2;
  cout << "Enter two numbers: " << endl;
  cin >> num1 >> num2;
  //skips spaces eats until spaces, tab, \n
  cin.ignore( INT_MAX, ‘\n’); //CRITICAL LINE
  cout<<”Enter name: “ << endl;
  string name; //automatically initialize to empty str
  getline(cin, name);
  //eats whole line including \n
  return 0;
}
```

OUTPUT FIXED NUM OF DECIMAL POINTS:

```
cout.setf( ios::showpoint );
//show decimal point even if not necessary
cout.setf( ios::fixed );
//make sure not print in scientific notification
cout.precision(2);
//use two digit after decimal point whether needed or not
//only affects way doubles are printed

cout.unsetf( ios::showpoint ); //undoes it
cout.unsetf( ios::fixed ); //undoes it
```

OUTPUT BOOLEAN IN ALPHA NUMERICS:

```
cout.setf( ios::boolalpha);
```

TYPE COMPATIBILITY:

Division: int/int = int (truncate decimal)
int/double = double/int = double/double = double

Cast int into double
double e = static_cast<double>(num);

PRECEDENCE RULE:

() → * / % → + - → =, +=, ….

ADD CHARS TO STRINGS:

```
string upper = "AWESOME";
string lower;
for(int i = 0; i < upper.length(); i++)
  lower += upper[i] + 32;


//output will be awesome in lower case
```

IF SYNTAX

```
if(…) {
   statement;
}else if(…){
   statement;
}else{
   statement;
}
```

REGULAR WHILE LOOP SYNTAX

```
while(…){
   statement;
}
```

DO WHILE LOOP SYNTAX

```
do{
  statement;
}while(…);
```

FOR LOOP SYNTAX

```
for( int k = 0; k < 10; k++){
   statement;
}
```

SWITCH CASE SYNTAX

```
 switch( type of variable ){
   case 0:
      statement;
      break;
   case 1:
   case 2:
      statement;
      break;
   default:
      statement;
}
```

Try not to declare variables
Switch type can be char, int, string as long as it is constant

STRING FUNCTIONS

```
       string s = "Hello";
       cout << s.size();   // writes 5 type size_t
       s = "Wow";
       cout << s.size();   // writes 3
       s = "";
       cout << s.size();   // writes 0

       string s = "Hello";      //  Hello
       cout << s.at(0);      // writes H
       cout << s[4];      // writes o
       cout << s[6];      // Undefined behavior!
       cout << s[-1];     // Undefined behavior!
       cout<<s.substr(1,3); //(starting index, length)
       //writes ell
#include <cctype>
```

isalpha(char c) //returns non-zero value if char is alphabet
isalnum(char c) //returns non-zero value if char is alphabet
isupper(char c) //returns non-zero value if char is upper case
islower(char c) //returns non-zero value if char is lower case
char a = tolower(char c) //turns char to lower case
char a = toupper(char c) //turns char to upper case

ASCII VALUES

‘0’ = 48	‘9’ = 57
‘A’ = 65	‘Z’ = 90
‘a’ = 97	‘z’ = 122

PASS BY VALUE

Declared as:
void foo(int i);   void foo(int);
Called as:
foo(12);   foo(‘A’);
void foo(int i){
  i = 12;
}

Nothing callee does affect caller’s variable
Not strict when taking in parameter

PASS BY REFERENCE

Declared as:
void foo(int **&** i);
Called as:
foo(i);   foo(j);
void foo(int & i){
  i = 12;
}

Callee can change value of caller’s variable
Function will only take in exact variable type requested
“actual thing” is sent

PASS BY CONST-REFERENCE

Declared as:
void foo(const int **&** i);
Called as:
foo(i);   foo(j);
void foo(const int & i){
  i = 12; ///will not build
}

Var are sent in a fixed way
Very strict and cannot be changed

OPTIONAL PARAMETERS FOR FUNCTIONS

Declared as:
void foo(int a, double d = 12.0); //d not required
Implement as:
void foo(int a, double d){
  statements…
}
RULE: all required parameters are grouped together and listed first

ARRAYS

int array[5]; //array of size 5, garbage value inside
int anotherArray[] = {1, 2, 3}; //array of size 3
int array2[5] ={1, 2}; // array of size 5, uninitialized values are set at zero

```
int array[5] = {1,2,3,4,5};
printArray(array, 5); //HAS TO PASS IN THE SIZE
void printArray(int array[], int size)
// array[ ] is an array parameter
{
   for(int i = 0; i < size; i++)
   {
     cout << array[i] << endl;
   }
}
void printArray(const int array[], int size);
//sends array as read only
```

BOOLEAN LAWS

De Morgan’s Law:
not (A or B) → (not A) && (not B)
not(A and B) → (not A) or (not B)

Types of Errors

1)Compilation Error: Syntax error, violate rules of the C++ language, compiler cannot create program
2)Logic Error: compiles, but things go wrong during run time

Covert String to Int

```
int convertInt(string s)
{
    int sum = 0;
    for(int k = 0; k<s.size();k++)
    {
        if(s[k] >= '0' && s[k] <= '9')
        {
            int num = s[k]-'0';
            sum= num + (sum*10);
        }
    }
    return sum;
}
```

Clip off first 6 characters of string

```
string t = "fingernail";


t = t.substr(6, t.size()-6);


// t is now "nail"
```

Make string to lower case:

```
string s = "Don't SHOUT!";
string t;
for (size_t k = 0; k !=s.size(); k++)
            t += tolower(s[k]);
cout << t;
//outputs “don’t shout!”
```

Copies non-letter into String

```
string s = "#1 in 2015: Yeah!";
string t;
for (size_t k = 0; k != s.size(); k++)
     if (!isalpha(s[k]))  // if not a letter
          t += s[k];       //   append it to t

// t is now "#1  2015: !"
```

Backwards String

```
int main() {
    cout << "Enter a phrase: ";
    string phrase;
    getline(cin, phrase);
    string backwards;
    for(int k = phrase.size()-1; k >=0 ; k--)
    {
        backwards += phrase[k];
    }
    cout << backwards;
}
```

print money sign

```
int main()
{
  int n = 40;

  for (int i = 0; i < n; i++)
  {
        for (int space = 1; space < (n - i); space++)
        {
              cout << " ";
        }
        for (int money = 0; money <= i; money++)
        {
              cout << "$";
        }
        cout << "|" << endl;
  }
```

Is Palindrome

```
int main() {
    cout << "Enter a palindrome: ";
    string phrase;
    getline(cin, phrase);

    int len = phrase.size();
    for (int i = 0; i < len / 2; i++)
    {
        int j = len - (i + 1);
        if (phrase[i] != phrase[j])
        {
            cout << "Not a palindrome" << endl;
            return 1;
        }
    }
    cout << "Is Palindrome";
}
```