Headers						Logic & Math Operators						Output Settings	Loops	
`#include<iostream>`						`+=`	`-=`	`*=`	`/=`	`||`	`&&`	`cin.ignore(1000,‘\n’)`	`for (strt;tst;cnt){}`	
`#include <string>`						**Variable Types**						`cout.precision(x)`	`do {} while (test)`	
`#include <cctype>`						`double`	`int`	`char`	`string`			`cout.setf(ios::fixed)`	`while (test) {}`	
`using namespace std`						`const type var_name`						**Switch**	**Chars**	
`int main() {}`						**I/O**						`switch (var) {}`	`s.size()`	`s[k]`
Comparative Operators						`cout << var`			`cin >> var`			`case x: action`	`isdigit()`	`isalpha()`
`==`	`!=`	`>`	`>=`	`<`	`<=`	`getline (cin, var)`						`break`	`isupper()`	`islower()`

Header:

```
#include<iostream>
#include<string>
#include<cctype>
using namespace std;
int main(){}
```

Variable initialization and input:

```
int count = 0;
cin >> count;
cin.ignore(10000,‘\n’);

string text = “”;
getline(cin,text);`
```

If, else statement:

```
int x = 0;
if (x == 0)
       cout <<  “x is 0”;
else
       cout << “x is not 0”;
```

Switch statement:

```
switch (x)
{
       case 1: cout << “A”;
              break;
       case 2: cout << “B”;
              break;
       case 3:
       case 4: cout << “C”;
              break;
       default: cout << “D”;
}
```

For loop:

```
for (int i = 0, i < 5, i++)
{}
```

Nested for loops:

```
for (int i = 0, i < 5, i++)
{
       for (int j = 0, j <= i,
j++)
       {}
}
```

While loop:

```
int i = 0
while (i < 5)
{
       i++;
}
```

Do-while loop:

```
int i = 0;
do
{
       i++;
} while (i < 5);
```

Variable types

`short`	Integer -32,768 to 32767
`int`	Integer -2.14M to 2.14M
`long`	Integer -2.14M to 2.14M
`float`	Number -10^{38} to 10^{38}
`double`	Number -10^{308} to 10^{308}
`char`	Single char in ‘’: ‘a’, ‘\n’, ‘&’
`bool`	true, false
`string`	Sequence of char in “”: “ab cd”

Headers

`#include<string>` to use strings.
`#include<cctype>` to use isalpha, etc.

Declaring / initializing a variable

Var names can only contain letters, numbers and underscores.
Var names cannot begin with numbers.
Example: a, text, count, _i, count_i

If var is declared within a branch of an if statement or within a loop, var can only be used within the statement.

Integer overflow is when value is too large to be represented by int, etc.

A char value cannot be empty.
`int  m = 5.6` will store 5.6 in m.
`int m = 11/5` will store 2 in m.
`int m = 2.6/0.5` will store 5 in m.

Operators

&& takes priority over ||.
! also applies to true/false functions.
Example: `!isalpha`

Flow Control

Parameter can only be int or char.
Common loop numbers:

Loop Parameters	Reps
`int i = 0; i <= 49; i++`	50
`int i = 0; i < 49, i++`	49
`int i = 0; i >= 0, i--`	50

The program will still compile with an infinite loop, however it will give a nonsensical answer.

Chars

For string “abcdefg hijk”:

- `s.size()` is 12
- `s[0]` is ‘a’
- `s[1]` is ‘b’
- `s[7]` is ‘ ’

In a loop, using `(int i = 0; i != s.size(); i++)` will end loop with string length number of repetitions.

Geometric patterns with nested loops

```
*
**
***
****

int max = 4;
for (int i=1; i<=max; i++)
{
       for (int j=1; j<=i;
j++)
       {
              cout << “*”;
       }
       cout << endl;
}

****
***
**
*

int max = 4;
for (int i=max; i>=1; i--)
{
       for (int j=1; j<=i;
j++)
       {
              cout << “*”;
       }
       cout << endl;
}
```

Common Errors

- ****MISSING SEMICOLON****
- Unmatched semicolons
- Unmatched quotes
- = instead of ==
- Division of integers (e.g. 5/2 = 2)
- Declaring var inside loop
- Using undeclared variable
- , instead of ; in for parameters
- Forgetting to break in switch
- Infinite loop (runtime error)
- Empty char (compilation error)
- Type mismatch (int x = 2.99)

String / Char Sequences

\n	New line
\t	Tab
\\\\	Backslash
\’	Single quote
\”	Double quote

Boolean Operators

```
if ( (score >= 0) && (score
<=10)
```

Is true if score is larger or equal to 0 and small or equal to 1

More cctype functions:
`toUpper()` to make character uppercase
`toLower()` to make character lowercase
+ operator can append two strings.

Reference Functions:
```
void addOne(int &y)
{
    y = y + 1;
}
```

Only used to pass integer / double arguments, not needed for strings / arrays.

Swap:
```
void swap(int& x, int& y)
{
    int t = x;
    x = y;
    y = t;
}
```

Arrays
Parameter in function is defined by var[]:
`int function(string a[])`
Calling a function with array parameter:
`function(a)`

Arrays can be declared like this:
`int foo[] = { 10, 20, 30 };`
Assumes size of 3.
`int bar [5] = { 10, 20, 30 };`
Empty elements are set to default values (normally 0).
Position values start from 0, for example:
`cout << bar[1]` prints out 20.

2D arrays: `foo[row][column]`
When passing a 2D array to a function, when declaring the array parameter
- Leave the first pair of square brackets empty
- Supply the actual declared size for the remaining dimensions

Bubble sort:
```
for (int i = 0; i < (n-1); i++) // Bubble
sort to sort in ascending order.
{
    string store = "";
    for (int k = 0; k < (n-i-1); k++)
        if (a[k] > a[k+1]) // Switch
neighbours if current is larger then next.
        {
            store = a[k];
            a[k] = a[k+1];
            a[k+1] = store;
        }
}
```

CStrings
Declaring strings as a sequence of characters.
“Hello” is a string literal, but can also be expressed by a sequence of 5 characters + the null terminator (\0) (6 elements for 5 letter word).

Declarations:
```
char myword[] = { 'H', 'e', 'l', 'l', 'o',
'\0' };
char myword[] = "Hello";
```

Arrays cannot be assigned values, following is invalid:
```
myword[] = "Bye";
```

User input:
`cin.getline(s,50)`

CString functions:
Using `#include <cstring>`:
`strcat(dest, source)`
Appends source to dest.

`strcpy(dest, source)`
Copies source to dest (replaces).

`strcmp(str1, str2)`
Compares str1 to str2.
If output is 0, str1 == str2.
If output is >0, str1 > str2.
If output is <0, str1 < str2.

`strlen(str)`
Outputs number of characters between beginning and null terminator.

An array of cstrings is declared by:
`char wordArray[100][10]`

Declaring array of cstrings as a parameter:
`int function(char words[][MAXWORDLEN+1])`

strcpy:
```
char s1[20];
char s2[20] = "Another new string";
strcpy(s1, ""); // Contents of s1 changed to
null string
strcpy(s1, "new string"); // Contents of s1
changed to "new string"
strcpy(s1, s2); // Contents of s1 changed to
"Another new string"
```

strcat:
```
char s1[20] = "Hello";
char s2[20] = "friend";
strcat(s1, ", my ");      // s1 now contains
"Hello, my "
strcat(s1, s2);           // s1 now contains
"Hello, my friend"
```