Manipulators

- Manipulators are operators used in C++ for formatting output. The data is manipulated by the programmer's choice of display.

endl manipulator

- endl is the line feed operator in C++. It acts as a stream manipulator whose purpose is to feed the whole line and then point the cursor to the beginning of the next line. We can use \n (\n is an escape sequence) instead of **endl** for the same purpose.

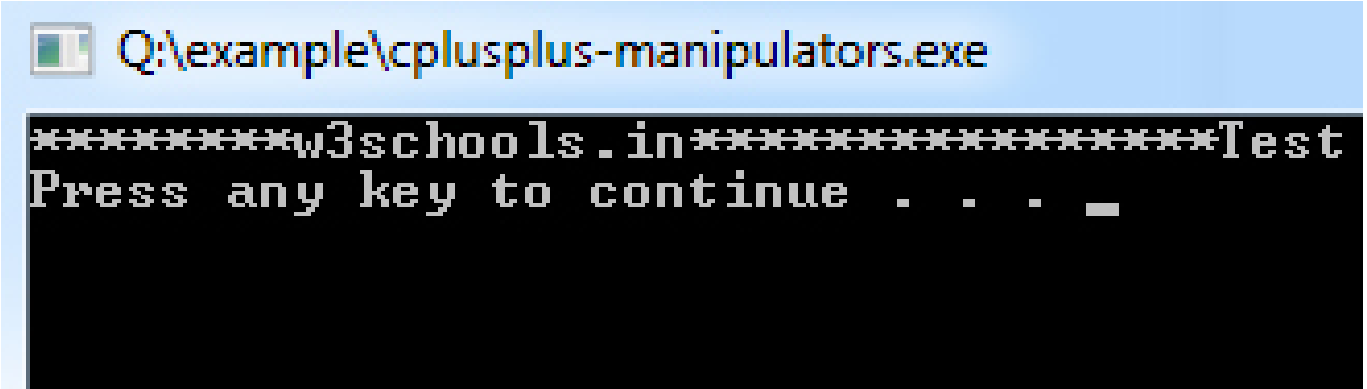
setw manipulator

- This manipulator sets the minimum field width on output.
- setw(x)

setfill manipulator

- This is used by the setw manipulator. If a value does not entirely fill a field, then the character specified in the setfill argument of the manipulator is used for filling the fields.

- #include <iostream>
- #include <iomanip>
- void main() {
- cout << setw(20) << setfill('*') << "w3schools.in" << setw(20) << setfill('*')<<"Test"<< endl;
- }



Type casting operator c++

- A cast is a special operator that forces one data type to be converted into another. As an operator, a cast is unary and has the same precedence as any other unary operator.
- The most general cast supported by most of the C++ compilers is as follows −
- (type) expression

Types:

- dynamic_cast <new_type> (expression)
  reinterpret_cast <new_type> (expression)
  static_cast <new_type> (expression)
  const_cast <new_type> (expression)

dynamic_cast

- dynamic_cast can be used only with pointers and references to objects. Its purpose is to ensure that the result of the type conversion is a valid complete object of the requested class.

```
class CBase { };
class CDerived: public CBase { };

CBase b; CBase* pb;
CDerived d; CDerived* pd;

pb = dynamic_cast<CBase*>(&d);      // ok: derived-to-base
pd = dynamic_cast<CDerived*>(&b);   // wrong: base-to-derived
```

static_cast

- static_cast can perform conversions between pointers to related classes, not only from the derived class to its base, but also from a base class to its derived.

```
class CBase {};
class CDerived: public CBase {};
CBase * a = new CBase;
CDerived * b = static_cast<CDerived*>(a);
```

reinterpret_cast

- reinterpret_cast converts any pointer type to any other pointer type, even of unrelated classes. The operation result is a simple binary copy of the value from one pointer to the other. All pointer conversions are allowed: neither the content pointed nor the pointer type itself is checked.

```
class A {};
class B {};
A * a = new A;
B * b = reinterpret_cast<B*>(a);
```

const_cast

- This type of casting manipulates the constness of an object, either to be set or to be removed. For example, in order to pass a const argument to a function that expects a non-constant parameter:

```
#include <iostream>
using namespace std;

void print (char * str)
{
  cout << str << endl;
}

int main () {
  const char * c = "sample text";
  print ( const_cast<char *> (c) );
  return 0;
}
```