I/O Format in C++

Default Format for integer, real, and character data:

- Integer:- Assume M=456 and N=5003

```
cout << M << N;
cout << M << " " << N;
```

- Real:- Assume X=1234.56789 and Y= 123456789.05520876

```
cout << X;
cout << Y;
```

- Character:- Assume ch="!" and str="String"

```
cout << ch << str << ch;
```

1

Two Types

The ios Flags

- **cout.setf(ios:flag)**
  ios:left
  ios:right
  ios:showpoint
  ios:fixed
  ios:scientific
- **cout.unsetf(ios:flag)**
- **cout.precision(n)**
- **cout.width(n)**
- **cout.fill(ch)**

Stream Manipulators

- **endl**
- **setw(N)**
- **setprecision(N)**

2

Parameter Passing in C++

Value Parameters (Call-by-value)

```
void swap(int x, int y)
{
  int temp;

  temp = x;
  x = y;
  y = temp;
}
Calling Block:
int u=3, v=5;
swap(u, v);
cout<<u<<" "<<v<<endl;
```

Reference Parameters (Call-by-reference)

```
void swap(int& x, int& y)
{
  int temp;

  temp = x;
  x = y;
  y = temp;
}
Calling Block:
int u=3, v=5;
swap(u, v);
cout<<u<<" "<<v<<endl;
```

3

Simulated Call-by-Reference in C

Value Parameters (Call-by-value)

```
void swap(int x,
          int y)
{
   int temp;

  temp = x;
  x = y;
  y = temp;
}
Calling Block:
int u=3, v=5;
swap(u, v);
cout<<u<<" "
    <<v<<endl;
```

Simulated Call-by-reference

```
void swap(int* x,
          int* y)
{
   int temp;

  temp = *x;
  *x = *y;
  *y = temp;
}
Calling Block:
int u=3, v=5;
swap(&u, &v);
cout<<u<<" "
    <<v<<endl;
```

Reference Parameters (Call-by-reference)

```
void swap(int& x,
          int& y)
{
   int temp;

  temp = x;
  x = y;
  y = temp;
}
Calling Block:
int u=3, v=5;
swap(u, v);
cout<<u<<" "
    <<v<<endl;
```

4

Constant Parameters in C++

- Reference parameters are sometimes used just for efficiency
  - although the parameter should not be modified in the body of a function!
- **const** keyword provides a protection mechanism for such formal parameters.

```
double power(const double x, int N)
{
  double val = 1.0;

  while (N--) val *= x;
  return val;
}
```

5

Default Parameters in C++

- A function's formal parameters can have default values:

```
void PrintParam(int N=5)
{
 cout << "Parameter is:"
      << N << endl;
}
```

Calling Program:

```
PrintParam(33);
PrintParam();
```

```
#include <iostream.h>
#include <iomanip.h>

void PrintInt(int A,
              int N=10)
{
   cout.unsef(ios.left);
   cout.setf(ios.right);
   cout<<setf(N)<<A<<endl;
}
```

Calling Program:

```
cout<<"Enter an integer:";
cin >> n;
cout<<"123456789012345";
PrintInt(n, 5);
PrintInt(n, 15);
PrintInt(n);
```

6