南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

C/C++ Program Design

Lab 2, data types and arithmetic operators

廖琪梅，于仕琪，王大兴

Formatting with cout

Floating-point types are displayed with a total of six digits, except that trailing zeros aren't displayed. The float number is displayed in *fixed-point notation* or else in *E notation* depending on the value of the number. In particular, *E notation* is used if the exponent is 6 or larger or -5 or smaller.

```
int main()
{
   double f1 = 1.200;
   std::cout << "f1 = " << f1 << std::endl;
   std::cout << "f1 + 1.0/9.0 = " << f1 + 1.0/9.0 << std::endl;

   double f2 = 1.67E2;
   std::cout << "f2 = " << f2 << std::endl;

   double f3 = f2 + 1.0/9.0;
   std::cout << "f3 = " << f3 << std::endl;
   std::cout << "f3 * 1.0e10 + 100 = " << f3 * 1.0e10 + 100 << std::endl;

   double f4 = 2.3e-4;
   std::cout << "f4 = " << f4 << std::endl;
   std::cout << "f4/10 = " << f4/10 << std::endl;

   return 0;
}
```

```
f1 = 1.2
f1 + 1.0/9.0 = 1.31111
f2 = 167
f3 = 167.111
f3 * 1.0e10 + 100 = 1.67111e+12
f4 = 0.00023
f4/10 = 2.3e-05
```

CC BY NC SA

C++ provides two methods to control the **output formats**

- ***Using member functions of ios class***
- ***Using iomanip manipulators***

- ***Using member functions of ios class***

1. cout.setf(): The setf() function has two prototypes, the first one is:
   cout.set(fmtflags);

std::ios_base::setf

```
fmtflags setf( fmtflags flags );                    (1)
fmtflags setf( fmtflags flags, fmtflags mask );     (2)
```

Formatting Constants

Constant	Meaning
`ios_base::boolalpha`	Input and output `bool` values as `true` and `false`.
`ios_base::showbase`	Use C++ base prefixes (0,0x) on output.
`ios_base::showpoint`	Show trailing decimal point.
`ios_base::uppercase`	Use uppercase letters for hex output, E notation.
`ios_base::showpos`	Use + before positive numbers.

cc BY NC SA

- ***Using member functions of ios class***

The second one is:

cout.set(fmtflags,fmtflags);

Arguments for `setf(long, long)`

Second Argument	First Argument	Meaning
`ios_base::basefield`	`ios_base::dec`	Use base 10.
	`ios_base::oct`	Use base 8.
	`ios_base::hex`	Use base 16.
`ios_base::floatfield`	`ios_base::fixed`	Use fixed-point notation.
	`ios_base::scientific`	Use scientific notation.
`ios_base::adjustfield`	`ios_base::left`	Use left-justification.
	`ios_base::right`	Use right-justification.
	`ios_base::internal`	Left-justify sign or base prefix, right-justify value.

cc BY NC SA

Using member functions of ios class

2. cout.width(len) //set the field width
3. cout.fill(ch) // fill character to be used with justified field
4. cout.precision(p) // set the precision of floating-point numbers

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

int main()
{
   cout << 56.8 << endl;
   cout.width(12);
   cout.fill('+');
   cout << 456.77 << endl;

   cout.precision(2);
   cout << 123.356 << endl;
   cout.precision(5);
   cout << 3897.678485 << endl;

   return 0;
}
```

```
56.8
++++++456.77
1.2e+02
3897.7
```

significant digits

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

int main()
{
   cout.setf(ios_base::fixed, ios_base::floatfield);
   cout << 56.8 << endl;
   cout.width(12);
   cout.fill('+');
   cout << 456.77 << endl;

   cout.precision(2);
   cout << 123.356 << endl;
   cout.precision(5);
   cout << 3897.678485 << endl;

   return 0;
}
```

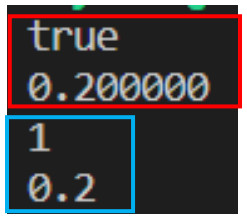
```
56.800000
++456.770000
123.36
3897.67848
```

precision of floating number

CC BY NC SA

The effect of calling ***setf()*** can be undone with ***unsef()***.

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

#include <iostream>
using namespace std;

int main()
{
    bool flag = true;
    float f = 0.20f;

    cout.setf(ios::showpoint);
    cout.setf(ios::boolalpha);
    cout << flag << endl;
    cout << f << endl;

    cout.unsetf(ios::boolalpha);
    cout.unsetf(ios::showpoint);
    cout << flag << endl;
    cout << f << endl;

    return 0;
}
```



CC BY NC SA

Standard Manipulators

C++ offers several manipulators to invoke setf(),automatically supplying the right arguments.

Some Standard Manipulators

Manipulator	Calls
boolalpha	setf(ios_base::boolalpha)
noboolalpha	unset(ios_base:: boolalpha)
showbase	setf(ios_base::showbase)
noshowbase	unsetf(ios_base::showbase)
showpoint	setf(ios_base::showpoint)
noshowpoint	unsetf(ios_base::showpoint)
showpos	setf(ios_base::showpos)
noshowpos	unsetf(ios_base::showpos)
uppercase	setf(ios_base::uppercase)
nouppercase	unsetf(ios_base::uppercase)

Manipulator	Calls
internal	setf(ios_base::internal, ios_base::adjustfield)
left	setf(ios_base::left, ios_base::adjustfield)
right	setf(ios_base::right, ios_base::adjustfield)
dec	setf(ios_base::dec, ios_base::base-field)
hex	setf(ios_base::hex, ios_base::base-field)
oct	setf(ios_base::oct, ios_base::base-field)
fixed	setf(ios_base::fixed, ios_base::floatfield)
scientific	setf(ios_base::scientific, ios_base::floatfield)

cc BY NC SA


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

int main()
{
    bool flag = false;
    double a = 2.3876;
    double b = 0.46e2;

    cout << boolalpha << flag << endl;
    cout << fixed << a << endl;
    cout << b << endl;

    cout << noboolalpha << flag << endl;
    cout.unsetf(ios::fixed);
    cout << a << endl;
    cout << b << endl;

    return 0;
}
```

```
false
2.387600
46.000000
0
2.3876
46
```

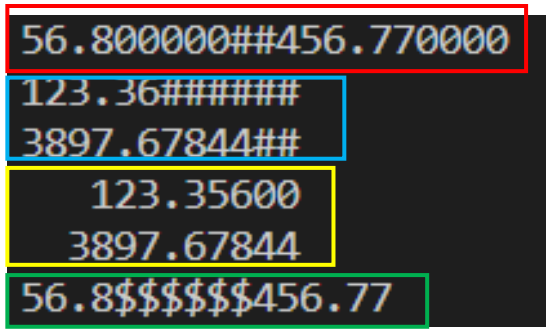
CC BY NC SA

Using iomanip manipulators

#include <iomanip>

1. setw(p) 2. setfill(ch) 3. setprecision(d)

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

int main()
{
  cout.setf(ios_base::fixed, ios_base::floatfield);
  cout << 56.8 << setw(12) << setfill('#') << 456.77 << endl;

  cout << left;
  cout << setw(12) << setprecision(2) << 123.356 << endl;
  cout << setw(12) << setprecision(5) << 3897.6784385 << endl;

  cout << right;
  cout << setw(12) << setfill(' ') << 123.356 << endl;
  cout << setw(12) << setfill(' ') << 3897.6784385 << endl;

  cout.unsetf(ios_base::fixed);
  cout << 56.8 << setw(12) << setfill('$') << 456.77 << endl;

  return 0;
}
```



CC BY NC SA

Type	Format Specifier
int	%d
char	%c
float	%f
double	%lf
short int	%hd
unsigned int	%u
long int	%li
long long int	%lli
unsigned long int	%lu
unsigned long long int	%llu
signed char	%c
unsigned char	%c
long double	%Lf

printf() vs cout

Which one do you prefer?

Example:

```
int a=1234;
float f=123.456;
char ch='a';
printf("%8d,%2d\n",a,a);
printf("%f,%8f,%8.1f,%.2f,%.2e\n",f,f,f,f,f);
printf("%3c\n",ch);
```

Sample output:

```
    1234,1234
123.456000,123.456000,    123.5,123.46,1.23e+02
   a
```

cc BY NC SA

Exercises

1.Compile and run the following program, what is the result?

You need to explain the reason to a SA to pass the test.

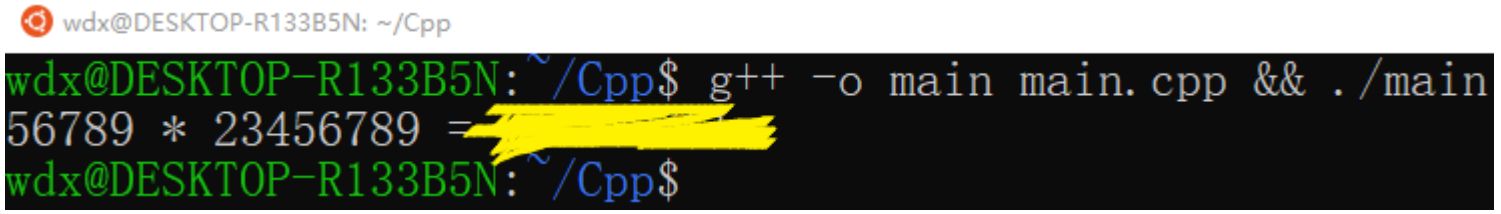
```
#include <stdio.h>

int main()
{
    signed char a = 127;
    unsigned char b = 0xff;
    unsigned char c = 0;

    a++;
    b++;
    c--;
    printf("a=%d\nb=%d\nc=%d\n",a,b,c);

    return 0;
}
```

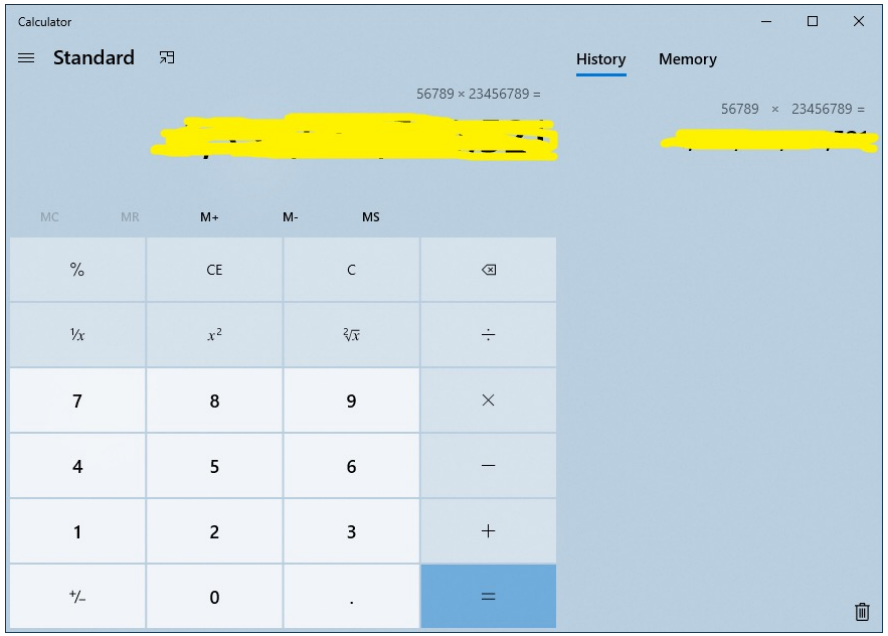
CC BY NC SA

Exercises

2. Write a program to calculate integer multiplication: 56789 * 23456789, and then print the result. Verify the result using a calculator.
If the result is wrong, what could be the reason? How to get the correct result for this exercise?

You need to explain the reason to a SA to pass the test.





CC BY NC SA

Exercises

3. Run the following source code and explain the result.

You need to explain the reason to a SA to pass the test.

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

int main()
{
    cout << fixed;
    float f1 = 1.0f;
    cout<<"f1 = "<<f1<<endl;

    float a = 0.1f;
    float f2 = a+a+a+a+a+a+a+a+a+a;
    cout<<"f2 = "<<f2<<endl;

    if(f1 == f2)
        cout << "f1 = f2" << endl;
    else
        cout << "f1 != f2" << endl;

    return 0;
}
```

cc BY NC SA

Exercises

4. Run the following source code and explain the result. Why the value of a and b are not equal? Explain the division operation with different types.

You need to explain the reason to a SA to pass the test.

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

int main()
{
    int a, b;
    double c, d;

    a = 19.99 + 21.99;
    b = (int)19.99 + (int)21.99;
    c = 23 / 8;
    d = 23 / 8.0;

    cout << "a = " << a << endl;
    cout << "b = " << b << endl;
    cout << "c = " << c << endl;
    cout << "d = " << d << endl;
    cout << "0/0= " << 0/0 << endl;

    return 0;

}
```

CC BY NC SA

Exercises

5. What is the output of the code as follows? What is the meaning of **auto** when defines a variable in C++?

You need to explain the reason to a SA to pass the test.

```
#include <iostream>

int main()
{
    auto a = 10;
    a = 20.5;
    a += 10.5;

    std::cout << a << std::endl;

    return 0;

}
```

CC BY NC SA