MA122 - Computer Programming and Applications

Indian Institute of Space Science and Technology

January 20, 2017

Lecture 5

The const qualifier

```
#include <iostream>
int main()
{

    const int months=12;;



    return 0;}
```

Lecture 5

1 const

2 float

3 Arithmetic Operators

Floating-point numbers

A floating-point number is composed of four elements:

- A sign: either negative or non-negative.
- A base (or radix): which expresses the different numbers that can be represented with a single digit (2 for binary, 10 for decimal, 16 for hexadecimal, and so on...).
- A significand (or mantissa): which is a series of digits of the aforementioned base. The number of digits in this series is what is known as precision.
- An exponent (also known as characteristic, or scale): which represents the offset of the significand, affecting the value in the following way:
  value of floating-point $=$ significand $\times$ base$^{\text{exponent}}$, with its corresponding sign.

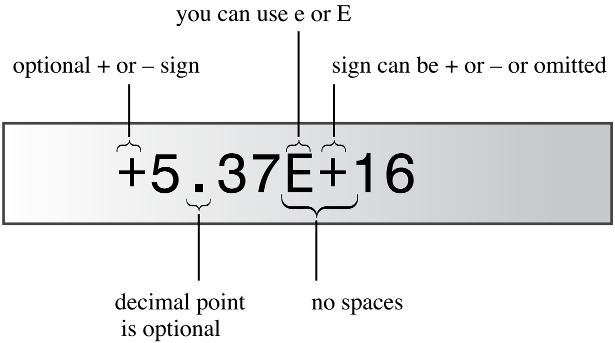
Writing floating-point numbers–first method

```
12.34              // floating-point
939001.32          // floating-point
0.00023            // floating-point
8.0                // still floating-point
```

Writing floating-point numbers–second method

```
2.52e+8             // can use E or e, + is optional
8.33E-4             // exponent can be negative
7E5                 // same as 7.0E+05
-18.32e13           // can have + or - sign in front
1.69e12             // 2010 Brazilian public debt in reais
5.98E24             // mass of earth in kilograms
9.11e-31            // mass of an electron in kilograms
```

E notation



you can use e or E
optional + or – sign
sign can be + or – or omitted
+5.37E+16
decimal point is optional
no spaces

cfloat/float.h

1) Number of decimal digits that are guaranteed to be preserved in text
2) Number of base RADIX digits that can be represented without losing precision

```
// the following are the minimum number of significant digits
#define DBL_DIG 15         // double
#define FLT_DIG 6          // float
#define LDBL_DIG 18        // long double

// the following are the number of bits used to represent the mantissa
#define DBL_MANT_DIG     53
#define FLT_MANT_DIG     24
#define LDBL_MANT_DIG    64

// the following are the maximum and minimum exponent values
#define DBL_MAX_10_EXP   +308
#define FLT_MAX_10_EXP   +38
#define LDBL_MAX_10_EXP  +4932
```

example

```
 #include <iostream>
int main()
{
  using namespace std;
  cout.setf(ios_base::fixed, ios_base::floatfield);
  float tub = 10.0 / 3.0;   // good to about 6 places

  double mint = 10.0 / 3.0; // good to about 15 places
  const float million = 1.0e6;

  cout << "tub = " << tub;
  cout << ", a million tubs = " << million * tub;
  cout << ",\nand ten million tubs = ";

  cout << 10 * million * tub << endl;
  cout << "mint=" << mint << "and a million mints= ";
  cout << million * mint <<endl;
  return 0; }
```

```
1.234f          // a float constant
2.45E20F        // a float constant
2.345324E28     // a double constant
2.2L            // a long double constant
```

precision problem

```cpp
// fltadd.cpp -- precision problems with float
#include <iostream>
int main()
{
  using namespace std;
  float a = 2.34E+22f;
  float b = a + 1.0f;
  cout << "a = " << a << endl;
  cout << "b - a = " << b - a << endl;
  return 0;
}
```

Lecture 5

1 const

2 float

3 Arithmetic Operators

precision problem

```cpp
// arith.cpp -- some C++ arithmetic
#include <iostream>
int main()
{
  using namespace std;
  float a, b;
  cout.setf(ios_base::fixed, ios_base::floatfield);
  cout << "Enter a number: ";
  cin >> a;
  cout << "Enter another number: ";
  cin >> b;
  cout << "a= " << a<< "; b = " << b << endl;
  cout << "a + b = " << a + b << endl;
  cout << "a - b = " << a - b << endl;
  cout << "a * b = " << a * b << endl;
  cout << "a / b = " << a/ b << endl;
  return 0; }
```

Division diversions

```
#include <iostream>
#include <iomanip>
int main()
{
  using namespace std;
  cout.setf(ios_base::fixed, ios_base::floatfield);
  cout << "Integer division: 9/5 = " << 9 / 5 << endl;

  cout << "Floating-point division: 9.0/5.0 = ";
  cout << 9.0 / 5.0 << endl;

  cout << "Mixed division: 9.0/5 = " << 9.0 / 5 << endl;
  cout << "double constants: 1e7/9.0 = ";
  cout << 1.e7 / 9.0 << endl;
```

Division diversions

```
  cout << "float constants: 1e7f/9.0f = ";
  cout << 1.e7f / 9.0f << endl;
    cout << setprecision(17);



    int f=383, m=3;
    double a;
    a=double(f)/m;


cout<< a<<endl;
  return 0;
}
```