南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

C/C++ Program Design

Lab 2, data types and arithmetic operators

王大兴，廖琪梅，于仕琪

Data types and arithmetic operators

- Commonly used data types (integers, floats, ...).
- Overflow
- Test integer range
- Conversion between char and integer
- Test float number precision

Commonly used data types: integer

- Declaring integers and doing simple arithmetic operations:

```
#include <iostream>

using std::cout;
using std::endl;

int main() {
    int a = 1234567890;
    int b = 1234567890;
    int sum = a+b;

    cout<<a<<" + "<<b<<" = "<<sum<<endl;

return 0;
}
```

Run the following command in your terminal and see what happens:

```
wdx@DESKTOP-R133B5N: ~/Cpp
wdx@DESKTOP-R133B5N:~/Cpp$ g++ -o main main.cpp && ./main
1234567890 + 1234567890 =
wdx@DESKTOP-R133B5N:~/Cpp$
```

Commonly used data types: float

- Declaring floating point numbers and doing simple arithmetic operations:

```
#include <iostream>

using std::cout;
using std::endl;

int main() {
    float a = 1234567.0;
    float b = 1.0;
    float sum = a+b;

    cout<<a<<" + "<<b<<" = "<<sum<<endl;

return 0;
}
```

Run the following command in your terminal and see what happens:

```
wdx@DESKTOP-R133B5N: ~/Cpp
wdx@DESKTOP-R133B5N:~/Cpp$ g++ -o main main.cpp && ./main
1.23457e+06 + 1 =
wdx@DESKTOP-R133B5N:~/Cpp$
```

Integer overflow

- Integers are stored in memory as a few number of bytes.
- The range of integers that can be properly represented is limited.
- Calculating numbers outside this range causes errors.

Conversion between char and integer

- The character type of c++ is char. Char and int can be converted.
- Characters are a type of integers in memory.

```
#include <cstdio>

int main() {
    char c = 'C';
    int i = c;

    printf("Convert '%c' to integer: %d\n", c, i );

return 0;
}
```

Run the following command in your terminal and see what happens:

```
wdx@DESKTOP-R133B5N: ~/Cpp
wdx@DESKTOP-R133B5N:~/Cpp$ g++ -o main main.cpp && ./main
Convert 'C' to integer:
wdx@DESKTOP-R133B5N:~/Cpp$
```

Conversion between char and integer

- You already know the integer value of character ‘C’, but why?
- ASCII table.

Floating point precision

- The range of floating point numbers is also limited.
- Floating point numbers also have precision problems.

```
#include <iostream>
using std::cout;
using std::endl;

int main() {
    float a = 0.1f;
    float sum = a+a+a+a+a+a+a+a+a+a;

    cout<<"sum: "<<sum<<endl;
    cout<<"sum equals to 1? "<<(sum==1.0)<<endl;

return 0;
}
```

Try the code on the left to see what happens.
Note that (sum==1.0) is a bool value, 1 means “true” and 0 means “false”.

wdx@DESKTOP-R133B5N: ~/Cpp

```
wdx@DESKTOP-R133B5N:~/Cpp$ g++ -o main main.cpp && ./main
sum: 1
sum equals to 1?
```

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(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.

● *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

coutset.cpp > main()

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

int main()
{
    cout.setf(ios_base::fixed, ios_base::floatfield);
    cout << 56.8;
    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;


}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
maydlee@LAPTOP-U1MO0N2F:/mnt/d/csourcecode/2021Spring/lab02$ g++ coutset.cpp
maydlee@LAPTOP-U1MO0N2F:/mnt/d/csourcecode/2021Spring/lab02$ ./a.out
56.800000++456.770000
123.36
3897.67848
maydlee@LAPTOP-U1MO0N2F:/mnt/d/csourcecode/2021Spring/lab02$
```

Using iomanip manipulators

#include <iomanip>

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

coutmanip.cpp > ...

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

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

    cout << setprecision(2) << 123.356 << endl;
    cout << setprecision(5) << 3897.6784385 << endl;

    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
maydlee@LAPTOP-U1MO0N2F:/mnt/d/csourcecode/2021Spring/lab02$ g++ coutmanip.cpp
maydlee@LAPTOP-U1MO0N2F:/mnt/d/csourcecode/2021Spring/lab02$ ./a.out
56.800000  456.770000
123.36
3897.67844
maydlee@LAPTOP-U1MO0N2F:/mnt/d/csourcecode/2021Spring/lab02$
```

cout.unsetf( )

coutunset.cpp > ...

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

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

    cout << setprecision(2) << 123.356 << endl;
    cout << setprecision(5) << 3897.6784385 << endl;

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

    cout << setprecision(2) << 123.356 << endl;
    cout << setprecision(5) << 3897.678385 << endl;

    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
maydlee@LAPTOP-U1MO0N2F:/mnt/d/csourcecode/2021Spring/lab02$ g++ coutunset.cpp
maydlee@LAPTOP-U1MO0N2F:/mnt/d/csourcecode/2021Spring/lab02$ ./a.out
56.800000  456.770000
123.36
3897.67844

56.8######456.77
1.2e+02
3897.7
maydlee@LAPTOP-U1MO0N2F:/mnt/d/csourcecode/2021Spring/lab02$
```

Exercises

1. **sizeof** operator returns the size, int bytes, of a type or a variable.

Compile the following program, what are the warnings? How to correct them?

```
#include <stdio.h>
#include <stdbool.h>
int main()
{
    printf("\nPrint size of the fundamental types:\n");
    printf("----------------------------------------\n");
    printf("The sizeof(char) is: %d bytes.\n", sizeof(char));
    printf("The sizeof(short) is: %d bytes.\n", sizeof(short));
    printf("The sizeof(int) is: %d bytes.\n", sizeof(int));
    printf("The sizeof(long) is: %d bytes.\n", sizeof(long));
    printf("The sizeof(long long) is: %d bytes.\n", sizeof(long long));
    printf("The sizeof(float) is: %d bytes.\n", sizeof(float));
    printf("The sizeof(double) is: %d bytes.\n", sizeof(double));
    printf("The sizeof(long double) is: %d bytes.\n", sizeof(long double));
    printf("The sizeof(bool) is: %d byte.\n", sizeof(bool));
    return 0;
}
```

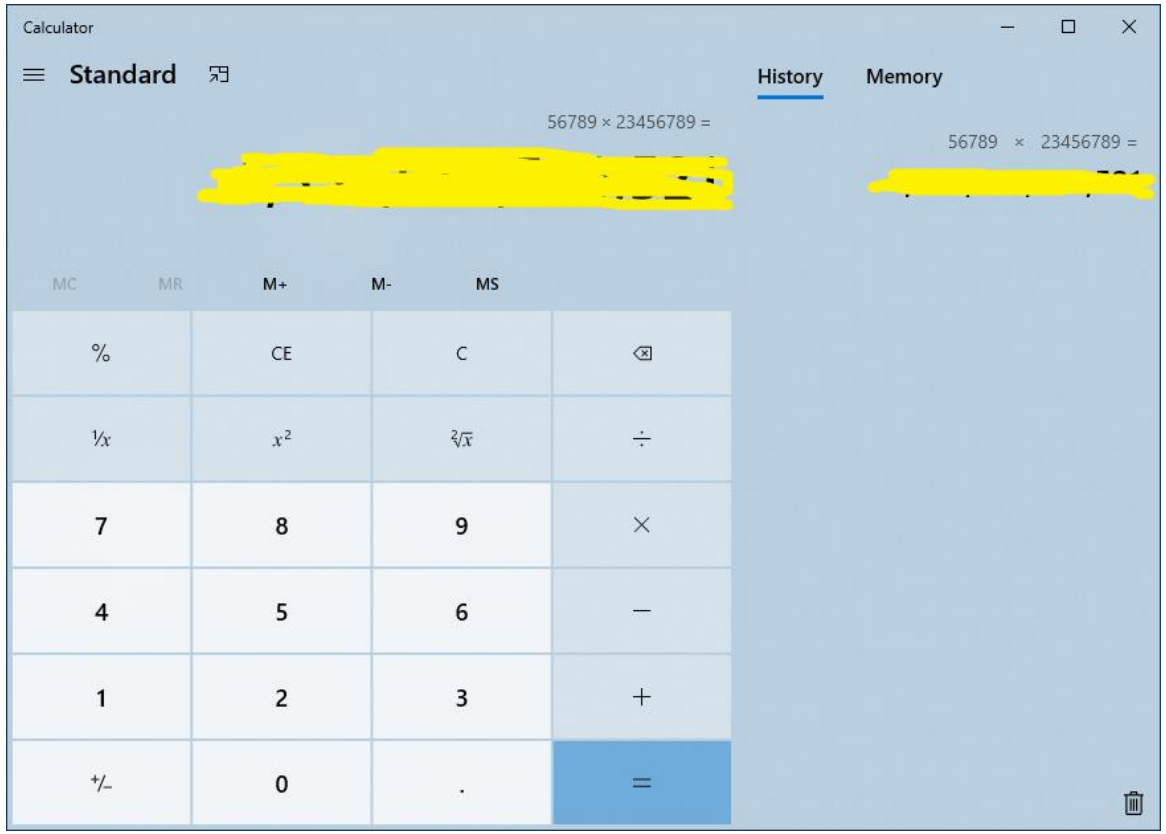
Exercises

2. Write a program to calculate integer multiplication: 56789 x 23456789, and then print the result. Find a method to get the correct answer if there is something wrong.

wdx@DESKTOP-R133B5N: ~/Cpp

```
wdx@DESKTOP-R133B5N:~/Cpp$ g++ -o main main.cpp && ./main
56789 * 23456789 =
wdx@DESKTOP-R133B5N:~/Cpp$
```

- Verify the result using a calculator.



- Congratulations if you get the right result!
- If the result is wrong, what could be the reason?

Exercises

3. Try conversions between **char** and **int**

- Choose some integers you like. Convert then to chars and print them.
- Or you can choose some chars and convert them to int.
- There are some special characters in the table, try some of them!

https://www.tutorialspoint.com/html/ascii_table_lookup.htm

Exercises

4. Run the following source code and find why the variable f2 is not 1.

```
void printFloat(float num)
{
    int inum = *(int*)&num;
    for (int i = 0; i < 32; ++i) {
        cout << ((inum&0x80000000) ? 1 : 0);
        if (i == 0 || i == 8)
            cout << " ";
        inum <<= 1;
    }
    cout << endl;
}
```

```
int main()
{
    float f1 = 1.0;
    cout<<"f1 = "<<f1<<endl;
    cout<<"The binary presentaion of f1"<<" is:"<<endl;
    printFloat(f1);
    cout<<endl;

    float a = 0.1f;
    float f2 = a+a+a+a+a+a+a+a+a+a;
    cout<<"f2 = "<<f2<<endl;
    cout<<"The binary presentaion of f2"<<" is:"<<endl;
    printFloat(f2);
    cout<<endl;

    return 0;
}
```