南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Advanced Programming

Lab 2, data types and arithmetic operators in C/C++

廖琪梅，于仕琪，王大兴, 王薇

Topics

- **1. Formatting with cout**
  - 1.1 *Using member functions of ios class*
  - 1.2 *Using iomanip manipulators*
- 2. **Debug** C/C++ by using gdb in VScode
- 3. **Data type conversions and calculations**
  - data storage, integer vs float
  - Integer promotions of Implicit conversions
- 4. **Practices**
  - data types and arithmetic operators

1. 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
```

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

1.1 Using member functions of ios class

1.2 Using iomanip manipulators

1.1 Using member functions of ios class

1.1.1 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.

1.1 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.

1.1 Using member functions of ios class

1.1.2. cout.**width**(len) //set the field width
1.1.3. cout.**fill**(ch) // fill character to be used with justified field
1.1.4. 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

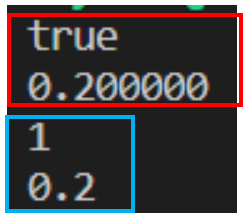
```
#include <iostream>
using namespace std;

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

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

```
#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;
}
```



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)


```cpp
#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
```

1.2 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;
}
```

```
56.800000##456.770000
123.36######
3897.67844##
   123.35600
  3897.67844
56.8$$$$$$456.77
```

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:

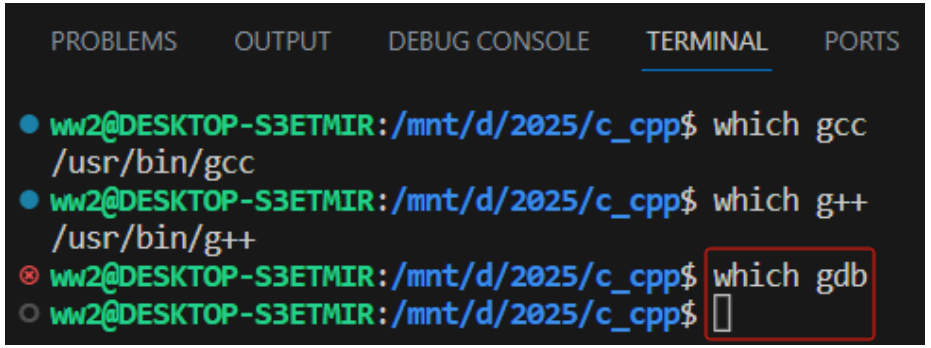
```
int a=1234;
float f=123.456;
char ch='a';
printf("%8d,%2d\n",a,a);
printf("%f,%8f,%8.1f,%.2f,%.2e\n",f,f,f,f,f);
printf("%3c\n",ch);
```

Sample output:

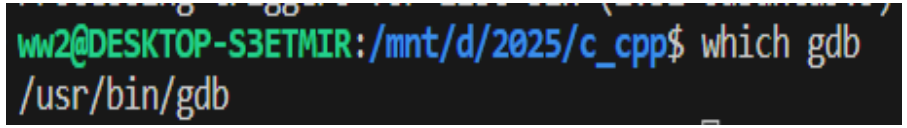
```
    1234,1234
123.456000,123.456000,    123.5,123.46,1.23e+02
   a
```

2. **Debug** C/C++ by using gdb in VScode

- 2.1 **Install "gdb" (the debug tool of C/C++)**
  - ➢ using cmd "**which gdb**" to check whether gdb is installed or no
    - ✓ if there is no info about gbd after running command "**which gdb**", it means that gdb is not installed, then
      - 1. using "**sudo apt undate**" to update package list
      - 2. using "**sudo apt install gdb**" to install gdb



    - ✓ If the installation directory of gdb is displayed after running command "**which gdb**" is executed, it means that gdb has been successfully installed.

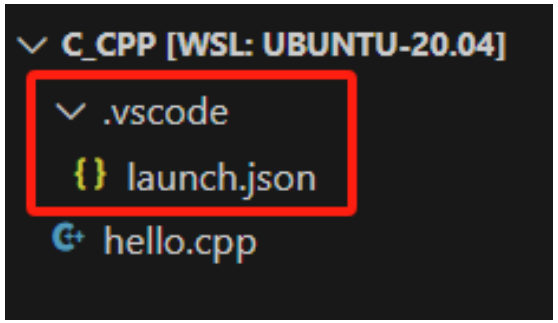


commands for install gdb

```
ww2@DESKTOP-S3ETMIR:/mnt/d/2025/c_cpp$ sudo apt update
[sudo] password for ww2:
Hit:1 http://archive.ubuntu.com/ubuntu focal InRelease
Get:2 http://archive.ubuntu.com/ubuntu focal-updates InRelease [128 kB]
Get:3 http://archive.ubuntu.com/ubuntu focal-backports InRelease [128 kB]
Get:4 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [3778 kB]
Get:5 http://security.ubuntu.com/ubuntu focal-security InRelease [128 kB]
Get:6 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages [3396 kB]
Get:7 http://security.ubuntu.com/ubuntu focal-security/main Translation-en [496 kB]
Get:8 http://security.ubuntu.com/ubuntu focal-security/restricted amd64 Packages [3406 kB]
Get:9 http://security.ubuntu.com/ubuntu focal-security/restricted Translation-en [477 kB]
Get:10 http://security.ubuntu.com/ubuntu focal-security/universe amd64 Packages [1032 kB]
Get:11 http://security.ubuntu.com/ubuntu focal-security/universe Translation-en [219 kB]
Get:12 http://archive.ubuntu.com/ubuntu focal-updates/main Translation-en [576 kB]
Get:13 http://archive.ubuntu.com/ubuntu focal-updates/restricted amd64 Packages [3553 kB]
Get:14 http://archive.ubuntu.com/ubuntu focal-updates/restricted Translation-en [497 kB]
Get:15 http://archive.ubuntu.com/ubuntu focal-updates/universe amd64 Packages [1254 kB]
Get:16 http://archive.ubuntu.com/ubuntu focal-updates/universe Translation-en [301 kB]
Fetched 19.4 MB in 6min 8s (52.7 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
165 packages can be upgraded. Run 'apt list --upgradable' to see them.
ww2@DESKTOP-S3ETMIR:/mnt/d/2025/c_cpp$ sudo apt install gdb
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  gdbserver libbabeltrace1 libc-dev-bin libc6 libc6-dbg libc6-dev libdw1 libelf1
Suggested packages:
  gdb-doc glibc-doc
The following NEW packages will be installed:
```

2. **Debug** C/C++ by using gdb in VScode

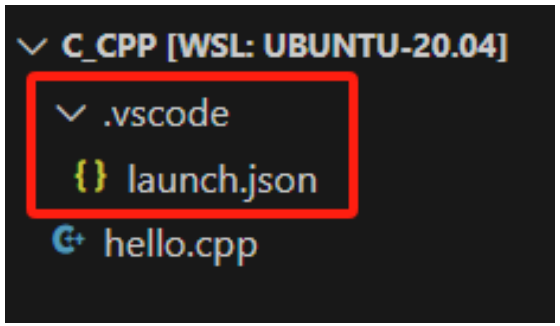
- 2.2 configure VSCode for using gdb to debug C/C++ code
  - ➢ create and edit ".vscode" folder and json files
    - ✓ step1. create a new folder named ".vscode" in the directory of C/C++ codes
    - ✓ step2. create a new json file named "launch.json" in the ".vscode" folder which is created in step1
      - edit "launch.json" to set gdb for debugging the execute file which is created by "g++ -g" / "gcc -g"
        - tips: option "-g" used with gcc/g++ is to generate information for debugging while compiling the C/C++ source code.



https://code.visualstudio.com/docs/cpp/config-linux


```
{
  "version": "0.2.0",
  "configurations": [
    {
      "name": "(gdb) Launch",
      "type": "cppdbg",
      "request": "launch",
      "program": "${fileDirname}/${fileBasenameNoExtension}",
      "args": [],
      "stopAtEntry": false,
      "cwd": "${workspaceFolder}",
      "environment": [],
      "externalConsole": false,
      "MIMode": "gdb",
      "setupCommands": [
        {
          "description": "Enable pretty-printing for gdb",
          "text": "-enable-pretty-printing",
          "ignoreFailures": true
        },
        {
          "description": "Set Disassembly Flavor to Intel",
          "text": "-gdb-set disassembly-flavor intel",
          "ignoreFailures": true
        }
      ]
    }
  ]
}
```

<--- An example of launch.json

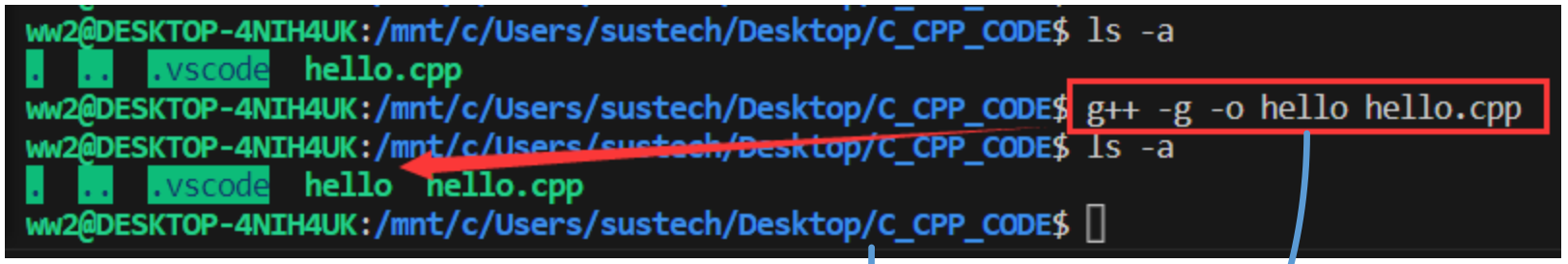


https://code.visualstudio.com/docs/cpp/config-linux

2. **Debug** C/C++ by using gdb in VScode

- 2.3 lunch gdb to debug in VS Code by “Run and Debug”

  ➢ compile the souce code with “**-g**” option to generate information for debug and generate the executable file

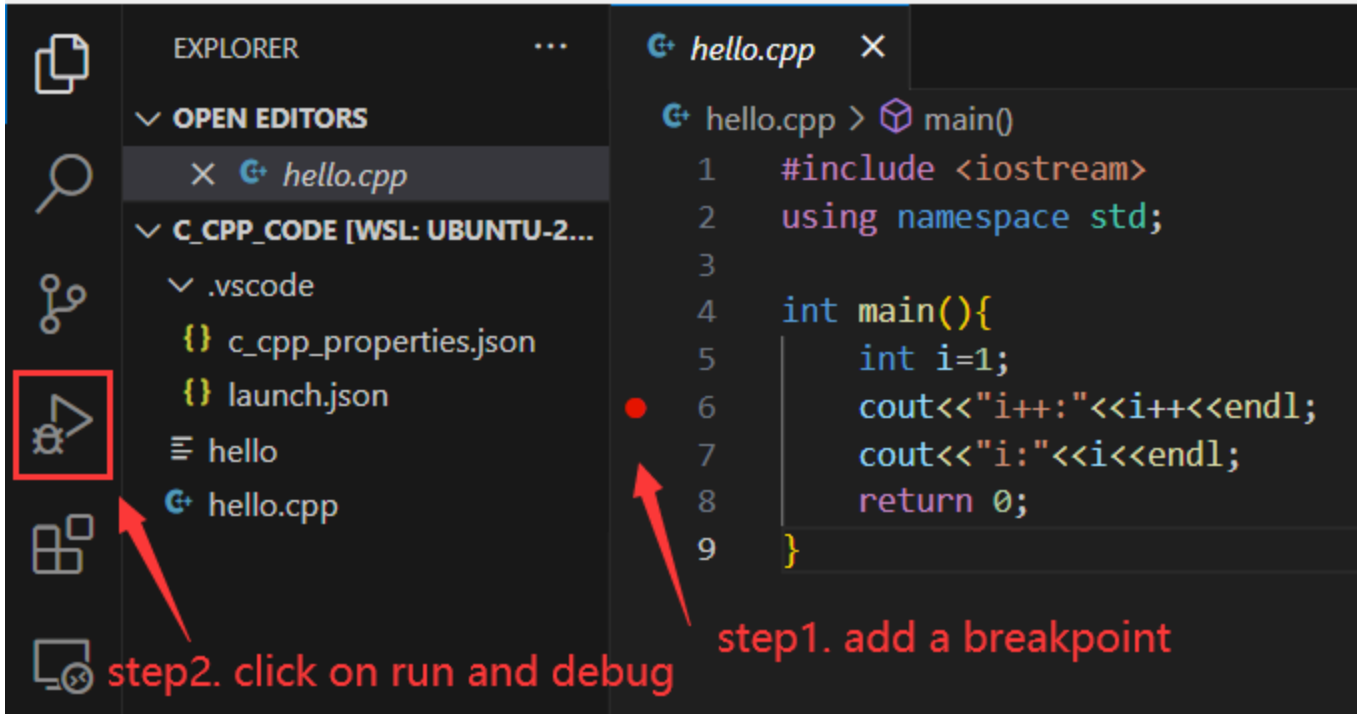


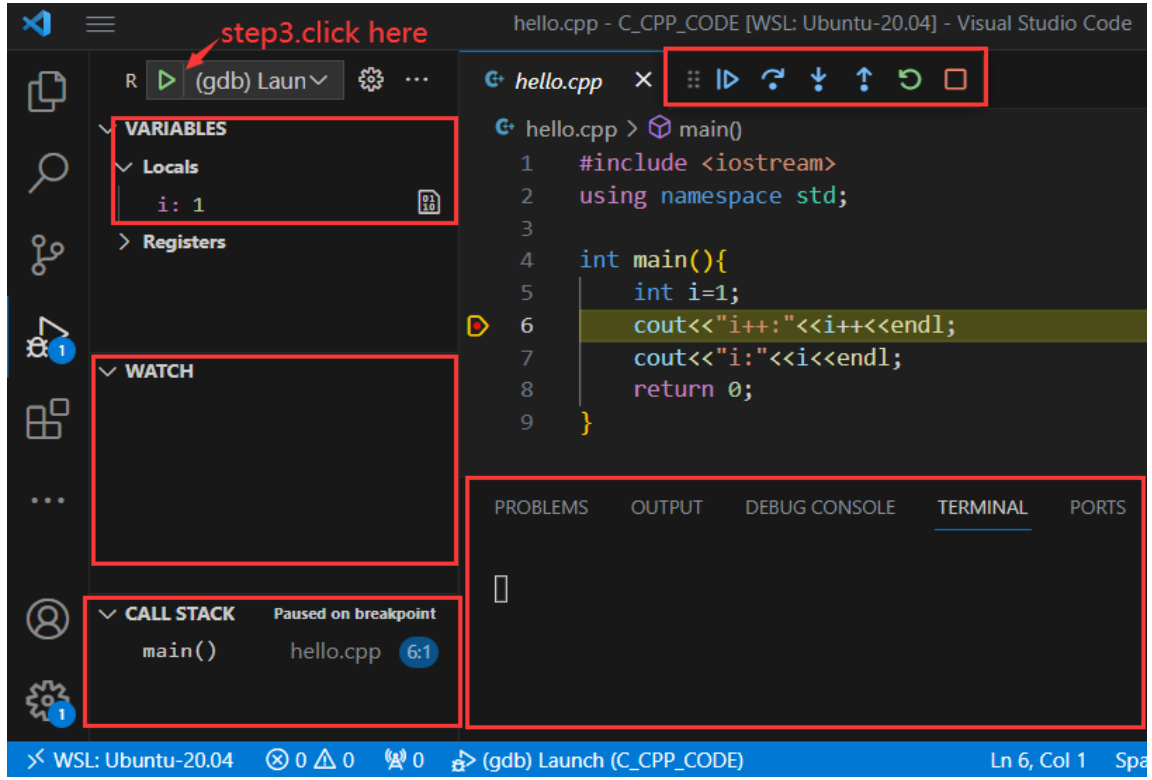
An example of launch.json-->

```
{
  "version": "0.2.0",
  "configurations": [
    {
      "name": "(gdb) Launch",
      "type": "cppdbg",
      "request": "launch",
      "program": "${fileDirname}/${fileBasenameNoExtension}",
      "args": [],
      "stopAtEntry": false,
      "cwd": "${workspaceFolder}",
```

2. **Debug** C/C++ by using gdb in VScode

- 2.4 set “breakpoint” on source file, lunch gdb to run and debug





https://code.visualstudio.com/docs/cpp/config-linux

2. **Debug** C/C++ by using gdb in VScode

- 2.5 View the data stored in a variable by gdb(optional)
  - ➢ During debugging, you can use GDB commands to view the data stored in variable(s).

✓step1. choose “DEBUG CONSOLE” window.

✓step2. run the command in command line in the “DEBUG CONSOLE” window.

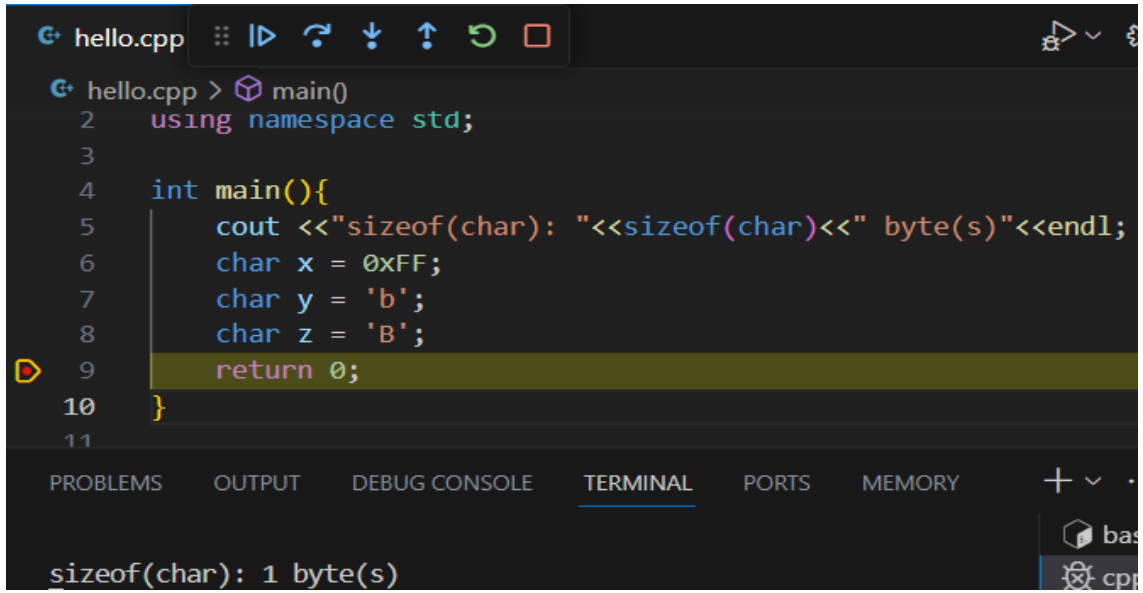
- -exec [gdb command] in vscode

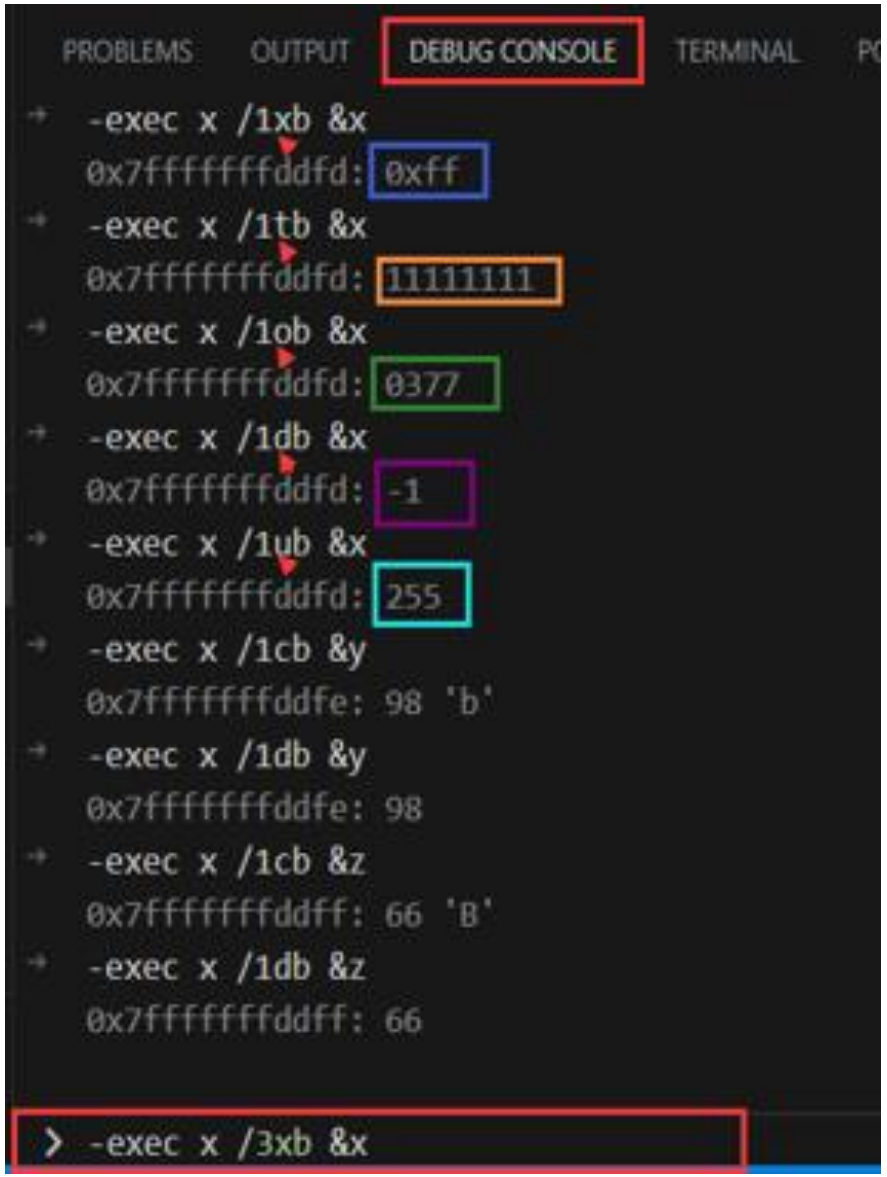
✓step3. View the results after executing the command in the “DEBUG CONSOLE” window.

2. **Debug** C/C++ by using **gdb** in VScode

- Using the command x (for “examine”) to examine memory in any of several formats, independently of your program’s data types.
  - ✓ x /nfu addr
    - n, the repeat count
    - f, the display format
    - u, the unit size

https://sourceware.org/gdb/current/onlinedocs/gdb.html/Memory.html#Memory





3. Data type conversions and calculations

- 3.1 data storage: integer vs float

lab3_2.cpp

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

int main(){

    int x=1;
    float y=1;
    cout<<"sizeof x: "<<sizeof(x)<<" byte(s), "
    <<"sizeof y: "<<sizeof(y)<<" byte(s)\n";
    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS MEMORY

```
ww2@DESKTOP-4NIH4UK:/mnt/c/Users/sustech/Desktop/C_CPP_CODE$ g++ -g -o lab3_2 lab3_2.cpp
ww2@DESKTOP-4NIH4UK:/mnt/c/Users/sustech/Desktop/C_CPP_CODE$
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
sizeof x: 4 byte(s), sizeof y: 4 byte(s)
```

lab3_2.cpp

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

int main(){

    int x=1;
    float y=1;
    cout<<"sizeof x: "<<sizeof(x)<<" byte(s), "
    <<"sizeof y: "<<sizeof(y)<<" byte(s)\n";
    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE endian

```
-exec x /1xw &x
0x7fffffffddf8: 0x00000001
-exec x /1xw &y
0x7fffffffddfc: 0x3f800000
-exec x /1xw &y
```

3. Data type conversions and calculations

- 3.2 Signed vs Unsigned
- Integer promotions of Implicit conversions

```
#include <stdio.h>

int main(){
    char x=0xff;
    unsigned char y=0xff;
    printf("x: 0x%x, %d , %u\n",x,x,x);
    printf("y: 0x%x, %d , %u\n",y,y,y);

    printf("x>>2: 0x%x, %d , %u\n",x>>2,x>>2,x>>2);
    printf("y>>2: 0x%x, %d , %u\n",y>>2,y>>2,y>>2);
    return 0;
}
```

```
x: 0xffffffff, -1 , 4294967295
y: 0xff, 255 , 255
x>>2: 0xffffffff, -1 , 4294967295
y>>2: 0x3f, 63 , 63
```

https://en.cppreference.com/w/c/language/conversion

4. Exercises

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

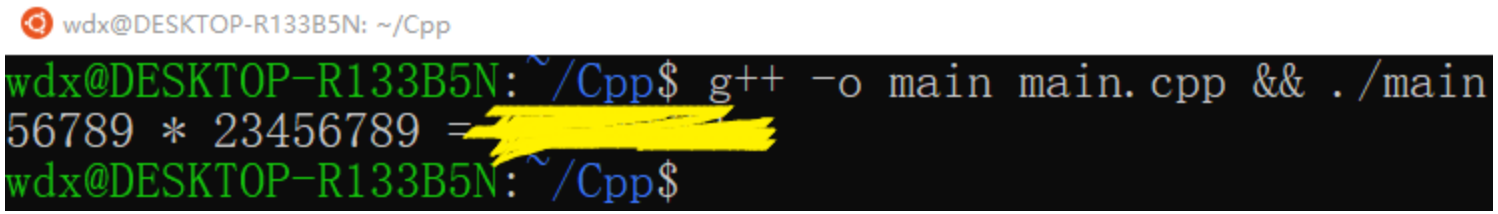
You need to explain the reason to a SA to pass the test.

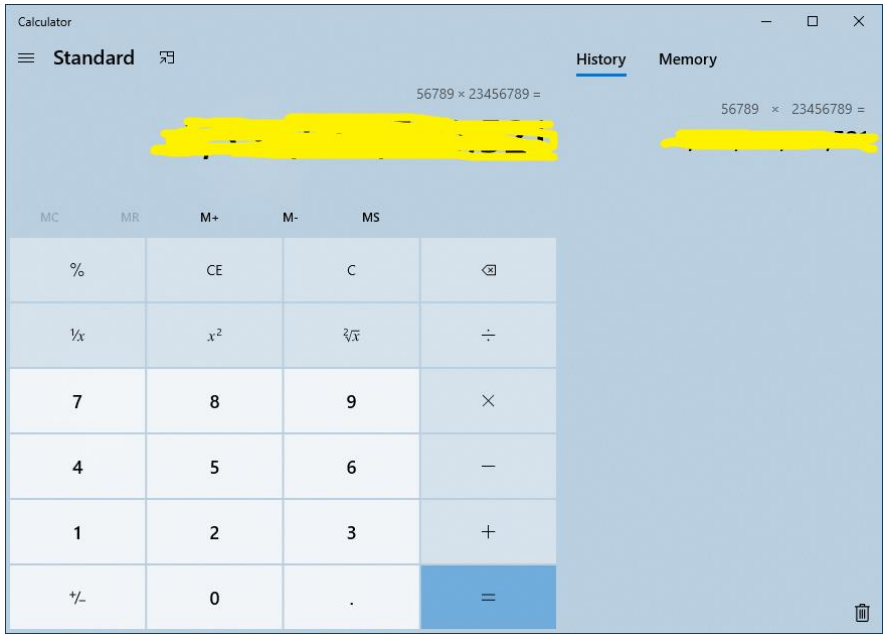
```
#include <stdio.h>
int main()
{
    signed char a = 127;
    unsigned char b = 0x7f;
    char c = 0x7f;

    a=a<<1;
    b=b<<1;
    c=c<<1;
printf("a=%x\nb=%x\nc=%x\n",a,b,c);
    printf("a=%d\nb=%d\nc=%d\n",a,b,c);
    a=a>>1;
    b=b>>1;
    c=c>>1;
printf("a=%x\nb=%x\nc=%x\n",a,b,c);
    printf("a=%d\nb=%d\nc=%d\n",a,b,c);

    return 0;
}
```

4. Exercises

4.2. 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.





4. Exercises

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

```
f1 = 1.000000
f2 = 1.000000
f1 != f2
```

```
#include <iostream>  //file name: lab3_p4_3.cpp
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)  //TIPS: Modify the code here
        cout << "f1 == f2" << endl;
    else
        cout << "f1 != f2" << endl;

    return 0;
}
```

Then using the method learnt in lecture2 to make the output of the code same as following picture .

```
f1 = 1.000000
f2 = 1.000000
f1 == f2
```

NOTE: DO NOT use if (f1=f2) instead of if(f1==f2).

4.Exercises

4.4. Complete the following source code to print the variables as the following picture 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,f,g;
    char h;

    a = 19.99 + 21.99;
    b = (int)19.99 + 21.99;
    c = 23 / 3;
    d = 23 / 3.0;
    f = 23 / 3.0e4;
    g = 23 / 3.0e5;
    h = 'b' - 32;

    //complete code here
    return 0;
}
```

```
B
41
40
7
7.66667
7.66667e-05
0.000766667
```

4.Exercises

4.5. 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;

    auto b=10.0;
    b = 20.5;
    b +=a;
    std::cout << b << std::endl;

    return 0;

}
```