C++ Programming

- Primary data types and variables

Outline

- Values
- Primary data types
- Operators
- Identifier
- Keywords
- Variables
- Declaration

Values and Data Types

Values

- There are different types of value, e.g.,
  - Age: `19`
  - Gender: `'m'` or `'f'`
  - Name: "`Tommy`"
  - Weight: `82.5`
  - Time: `13:25:16`

Values

- There are different types of value, e.g.,
  - Age: `19` (integer)
  - Gender: `'m'` or `'f'` (char)
  - Weight: `82.5` (float)
  - Name: `"Tommy"` (string)
  - Time: `13:25:16` (structure)

Values

- There are different types of value, e.g.,
  - Age: `19` (integer)
  - Gender: `'m'` or `'f'` (char)
  - Weight: `82.5` (float)
  - Name: "`Tommy`" (string)
  - Time: `13:25:16` (structure)

primary data types

Primary Data Types

- `int`
  - integer number.
  - The biggest integer that can be expressed in acomputer depends on the host computer (32 bits or 64 bits)
- `char`
  - single characters
  - Each character has an ASCII code
- `float`
  - Real number (single precision float point)
- `double`
  - Real number (double precision float point)

Primary Data Types

- `int`
  - integer number.
  - The biggest integer that can be expressed in acomputer depends on the host computer (32 bits or 64 bits)
- `char`
  - single characters
  - Each character has an ASCII code
- `float`
  - Real number (single precision float point)
- `double`
  - Real number (double precision float point)

Primary Data Type – `int`

- `int`
  - Anatural number (including 0), a negative number
  - 4 bytes
  - E.g., `10, 20, 10000`
  - Can be expressed in
    - Decimal (base-10) : 12
    - Binary (base-2)
    - Hexadecimal (base-16): 0x12
    - Octal (base-8): 011

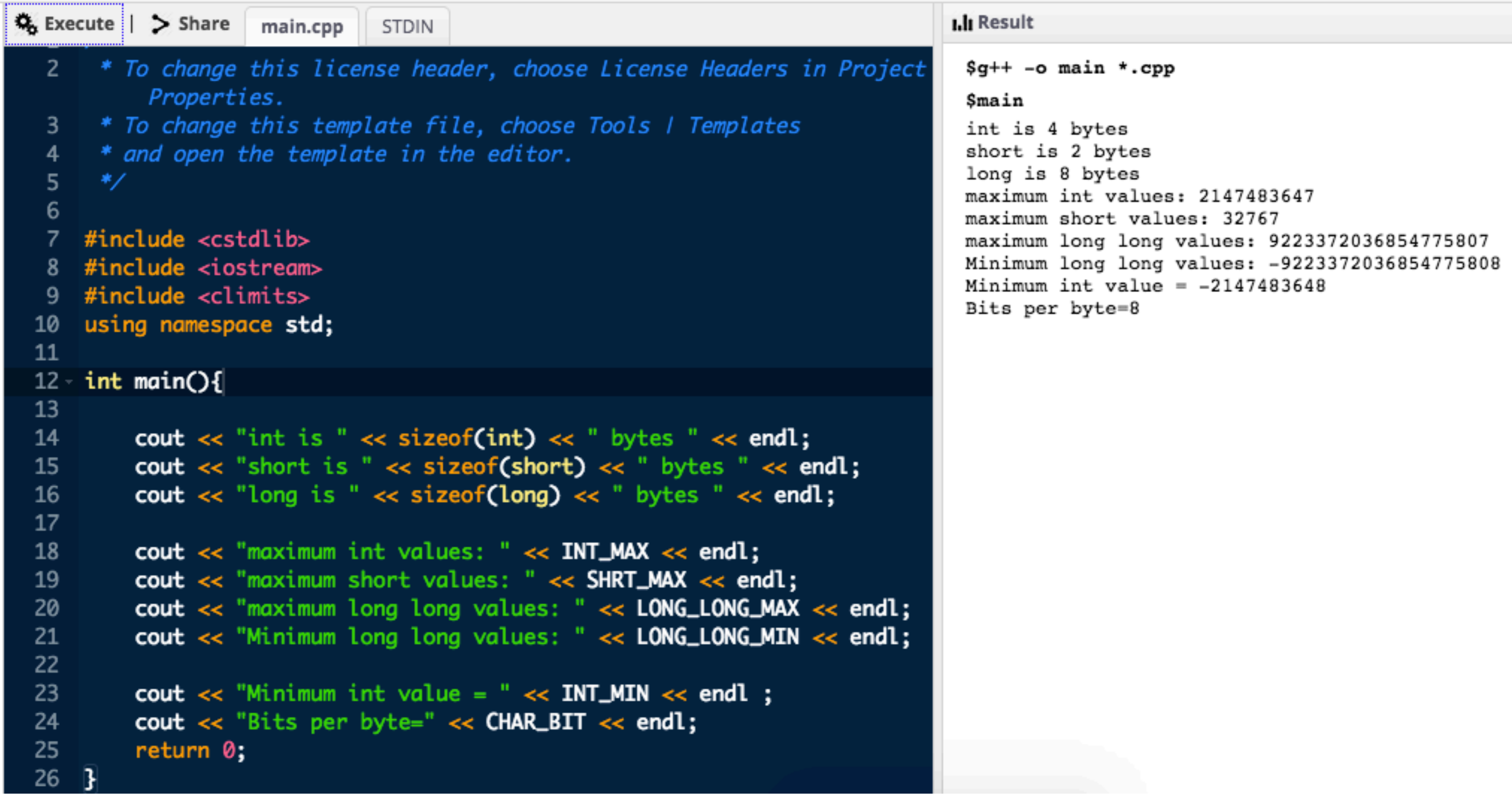
Primary Data Type – `int`

- Other `int` types
  - `short int (short)`
    - 2 bytes, $-2^{15} \sim 2^{15} - 1$
    - E.g., `12, 20`
  - `long int(long)`
    - 4 bytes, $-2^{31} \sim 2^{31} - 1$
    - `20, 20`**`L`**`, -2000, `**`0x`**`ffff`**`L`**

Primary Data Type – `int`

- Other `int` types
  - `unsigned int`
    - 4 bytes, $0 \sim 2^{32} - 1$
    - E.g., `20, 12u, 0xffu`
  - `long long int`
    - 8 bytes, $-2^{63} \sim 2^{63} - 1$
    - E.g., `20, 20LL`

Example



```cpp
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

#include <cstdlib>
#include <iostream>
#include <climits>
using namespace std;

int main(){

    cout << "int is " << sizeof(int) << " bytes " << endl;
    cout << "short is " << sizeof(short) << " bytes " << endl;
    cout << "long is " << sizeof(long) << " bytes " << endl;

    cout << "maximum int values: " << INT_MAX << endl;
    cout << "maximum short values: " << SHRT_MAX << endl;
    cout << "maximum long long values: " << LONG_LONG_MAX << endl;
    cout << "Minimum long long values: " << LONG_LONG_MIN << endl;

    cout << "Minimum int value = " << INT_MIN << endl ;
    cout << "Bits per byte=" << CHAR_BIT << endl;
    return 0;
}
```

```
$g++ -o main *.cpp
$main
int is 4 bytes
short is 2 bytes
long is 8 bytes
maximum int values: 2147483647
maximum short values: 32767
maximum long long values: 9223372036854775807
Minimum long long values: -9223372036854775808
Minimum int value = -2147483648
Bits per byte=8
```

Primary Data Types

- `int`
  - integer number.
  - The biggest integer that can be expressed in acomputer depends on the host computer (32 bits or 64 bits)
- `char`
  - single characters
  - Each character has an ASCII code
- `float`
  - Real number (single precision float point)
- `double`
  - Real number (double precision float point)

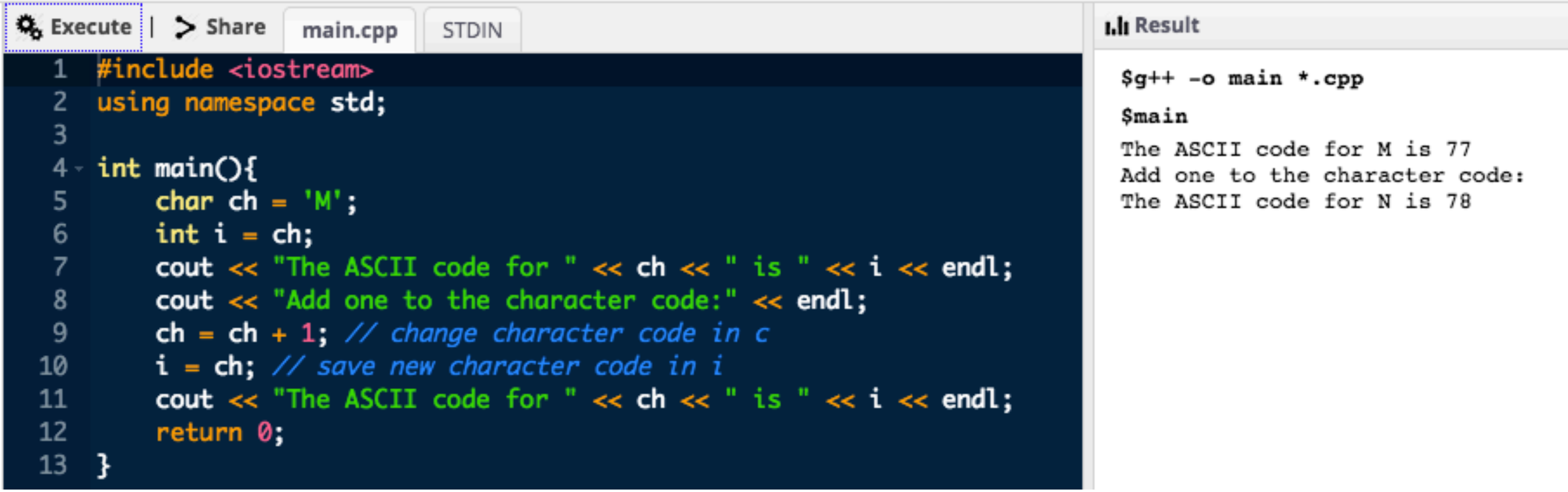
Primary Data Type – `char`

- `char`
  - 1 byte
  - `-128 ~ 127`
  - E.g., `'a', '1', '+'`
- Attention
  - `'1'` is different from 1
  - `'+'` is different from +
  - `'a'` is different from `a`
  - `'a'` is different from `"a"`
- Every char corresponds to an integer code

Char – ASCII

- ASCII
  - American Standard Code for Information Interchange
  - Tables
    - http://www.ascii-code.com/

Example



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

int main(){
    char ch = 'M';
    int i = ch;
    cout << "The ASCII code for " << ch << " is " << i << endl;
    cout << "Add one to the character code:" << endl;
    ch = ch + 1; // change character code in c
    i = ch; // save new character code in i
    cout << "The ASCII code for " << ch << " is " << i << endl;
    return 0;
}
```

```
$g++ -o main *.cpp

$main

The ASCII code for M is 77
Add one to the character code:
The ASCII code for N is 78
```

Primary Data Types

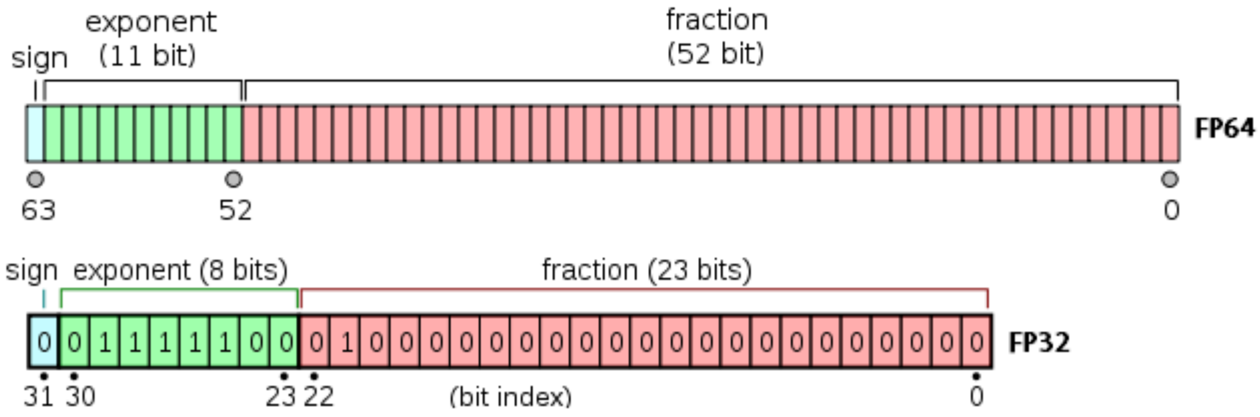
- `int`
  - integer number.
  - The biggest integer that can be expressed in acomputer depends on the host computer (32 bits or 64 bits)
- `char`
  - single characters
  - Each character has an ASCII code
- `float`
  - Real number (single precision float point)
- `double`
  - Real number (double precision float point)

Primary Data Type – `float, double`

- Float
  - 4 bytes
  - E.g., `1.2, 2.5e8` (Scientific notation. 2.5×10^8)
  - Range: `3.4e-38 ~ 3.4e+38 (absolute value)`
- Double
  - 8 bytes
  - Range: `1.7e-308 ~ 1.7e+308 (absolute value)`

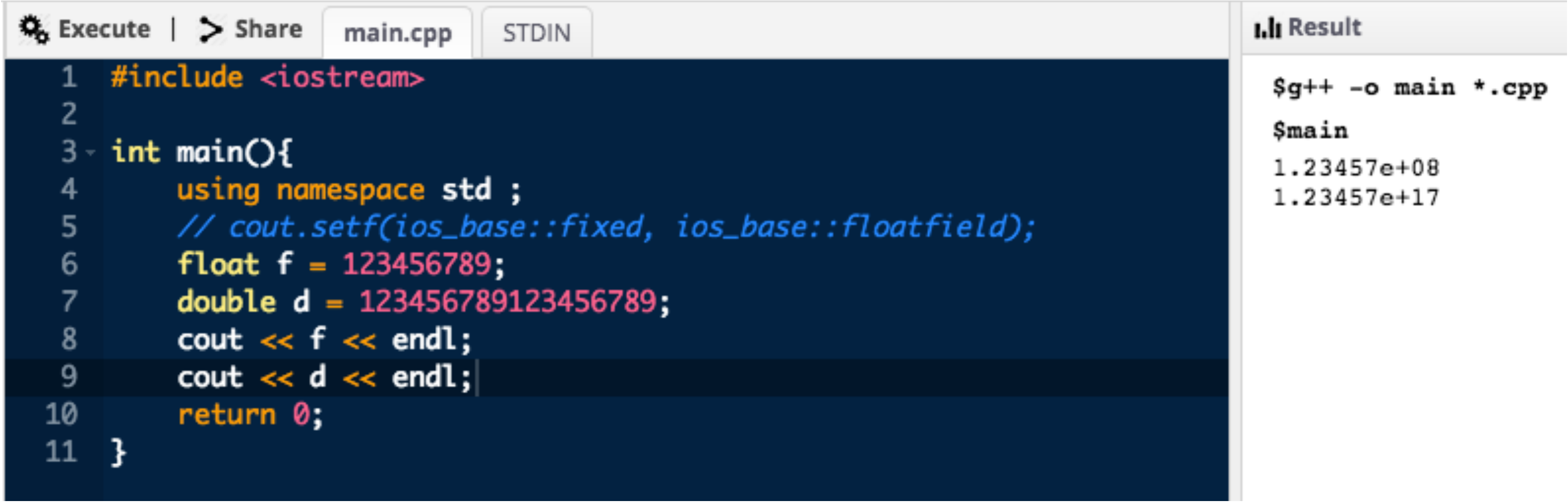
Single precision V.S. Double precision

- Single precision is the 32 bit representation of numerical values in computers.
- Double precision uses 64 bits to represent a value.



Float v.s. Double

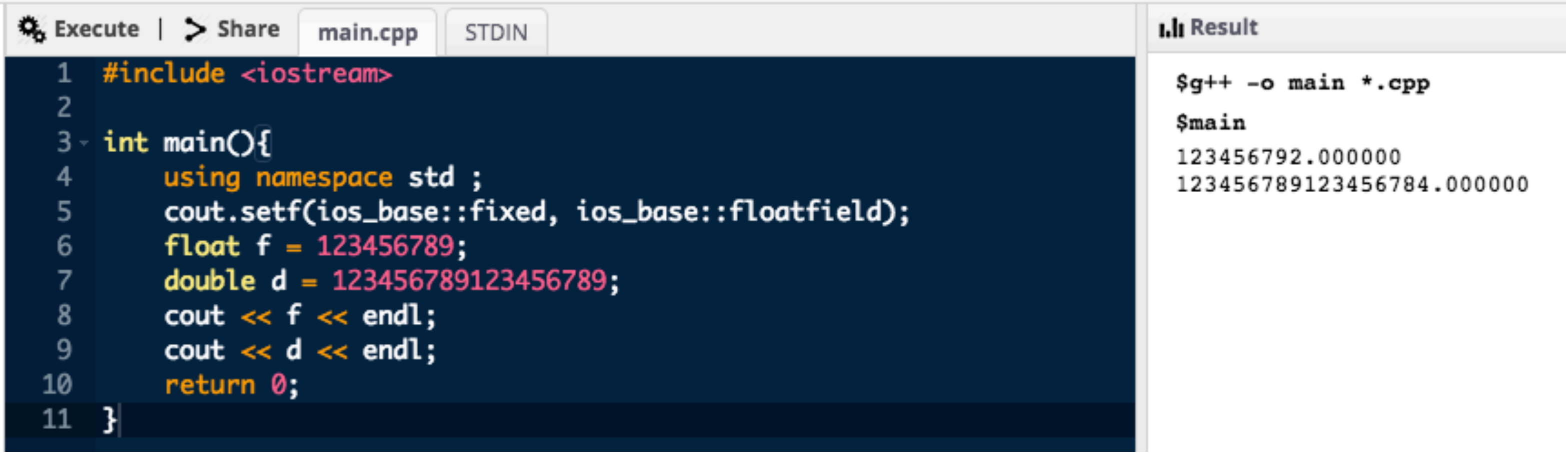
- **Double** is more widely ranged and more accurate than **float.**
- **a double has 2x the precision of float. In general a double has 15 decimal digits**
- **of precision, while float has 7.**



```cpp
#include <iostream>

int main(){
    using namespace std ;
    // cout.setf(ios_base::fixed, ios_base::floatfield);
    float f = 123456789;
    double d = 123456789123456789;
    cout << f << endl;
    cout << d << endl;
    return 0;
}
```

```
$g++ -o main *.cpp
$main
1.23457e+08
1.23457e+17
```

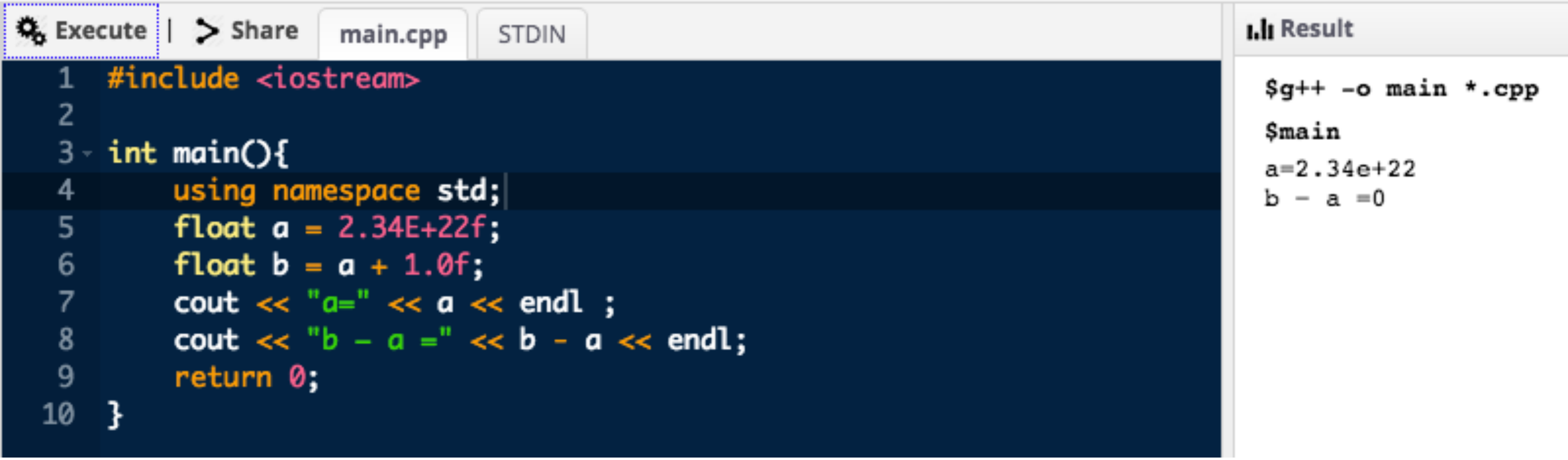


```cpp
#include <iostream>

int main(){
    using namespace std ;
    cout.setf(ios_base::fixed, ios_base::floatfield);
    float f = 123456789;
    double d = 123456789123456789;
    cout << f << endl;
    cout << d << endl;
    return 0;
}
```

```
$g++ -o main *.cpp
$main
123456792.000000
123456789123456784.000000
```

Float v.s. Double



```cpp
#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;
}
```

```
$g++ -o main *.cpp
$main
a=2.34e+22
b - a =0
```

- We expect mathematically to get a value of 1. Problem float has only represents the first 6 or 7 digits in a number.

Floating Point Numbers

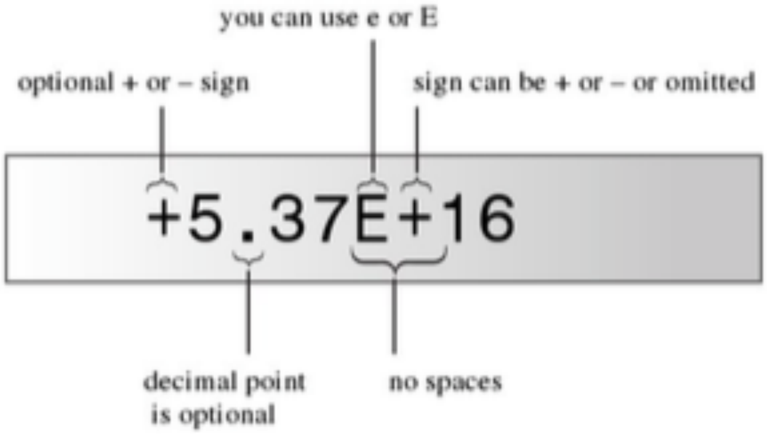
C++ has two ways of writing floating point numbers

- Standard decimal notation, e.g. 8.01
- E notation, for very big and very small numbers. e.g. float million = 1.0e6

E Notation

A quick review of scientific notation.

FIGURE 3.3
E notation.



```
2.52e+9 /* Valid */          2.52 e+9 /* Not Valid */
2.52E9  /* Valid */
```

Floating Point Numbers

- By default when you use a floating-point constant in a program it is a double.
- You can change to float or long double in the following manner:

```
1.234        // a double
1.234f       // a float
1.23e20F     // a float
3.221E28     // a double constant
2.2L         // a long double
```

Boolean Type

The bool type named after English mathematician George Boole who developed a mathematical notation for the laws of logic.

true

false

Variable

Class Exercises

Guess what the output of this program is?

```
#include<iostream>
using namespace std;
int main()
{
   int value1, value2, sum;
   value1 = 50;
   value2 = 25;
   sum = value1 + value2;
   cout << value1 << '+' << value2 << '=' << sum << endl;
   return 0;
}
```

Remember this page??

Sum.cpp

Class Exercises

Guess what the output of this program is?

```
#include<iostream>
using namespace std;
int main()
{
   int value1, value2, sum;
   value1 = 50;
   value2 = 25;
   sum = value1 + value2;
   cout << value1 << '+' << value2 << '=' << sum << endl;
   return 0;
}
```

Values

???

Sum.cpp

Identifiers (Variable Names)

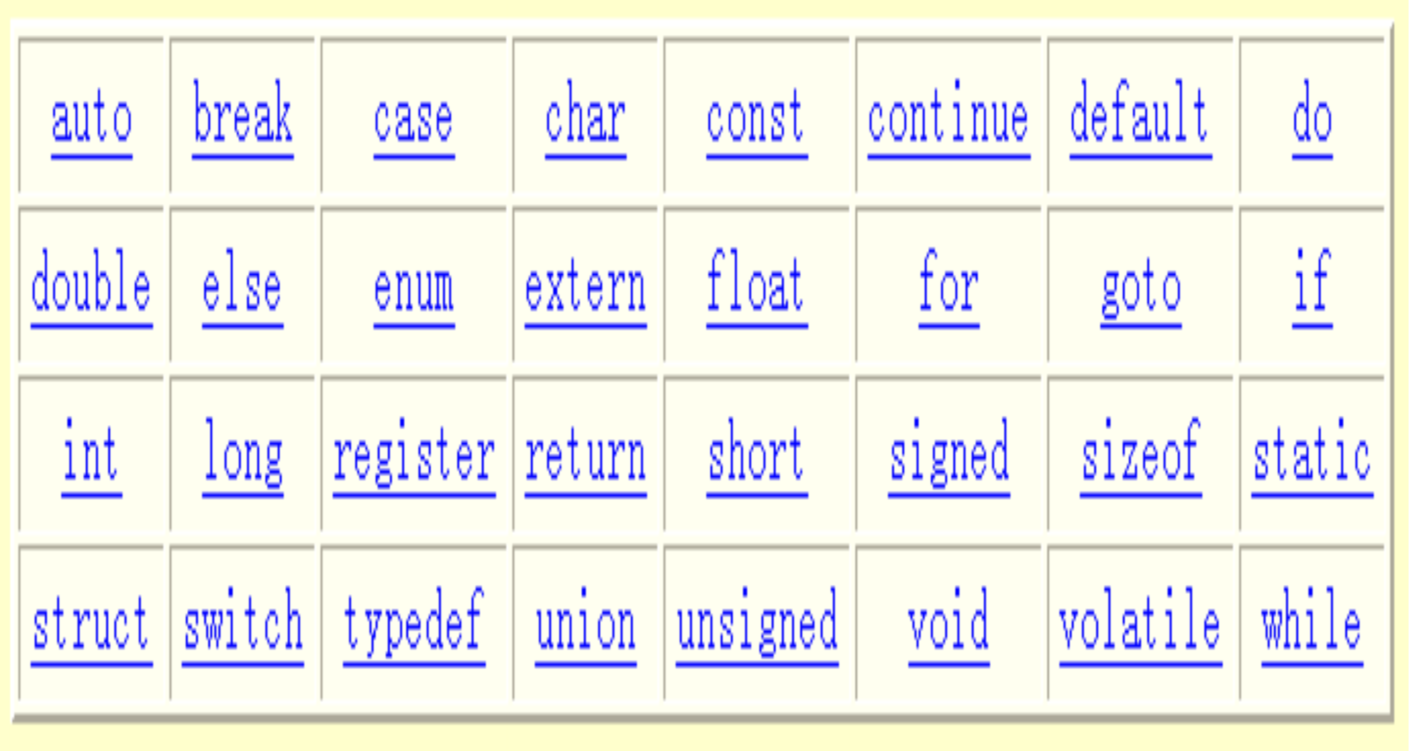
- An identifier consists of a letter or underscore followed by any sequence of letters, digits or underscores
  - E.g., _Is, This_Is, A12, a23
- Identifiers are case-sensitive!
  - Hello, hello,
  - whoami, whoAMI, WhoAmI
  - C, c

Are they same?

Identifiers (Variable Names)

- Identifiers cannot have special characters in them
  - E.g., `X=Y, J-20, #007` are invalid identifiers.
- C++ keywords (reserved words) cannot be used as identifiers.
  - Keywords
    - `int, float, double, short, char, class,`
    - more in the next page …..
  - Keywords have been given special meanings in C++.
- Choose identifiers that are meaningful and easy to remember.

Keywords



auto	break	case	char	const	continue	default	do
double	else	enum	extern	float	for	goto	if
int	long	register	return	short	signed	sizeof	static
struct	switch	typedef	union	unsigned	void	volatile	while

Class Exercises

- Are these the valid variable names?
  - `_123`
  - `_abc`
  - `Example`
  - `Abc123`
  - `unsigned`
  - `int`
  - `a%b`
  - `2example`
  - `Xx`

Class Exercises

- Are these the meaningful variable names
  - `a`
  - `abc`
  - `sum`
  - `product`
  - `numberOfApples`
  - `nApples`
  - `X_Value`
  - `Y_Value`
  - `price`

Variable Declaration and Assignment

Declaration and Assignment

- Every variable used in a program must declare its type before it is used
  - Declaration format
    - `TYPE variable_name_list;`
  - E.g.,
    - `int i;`
    - `float f;`
    - `double area;`
    - `unsigned int number;`
    - `int number, index, grade;`
    - `bool b = true;`
- Avariable name can be declared only once in a pair of brackets.

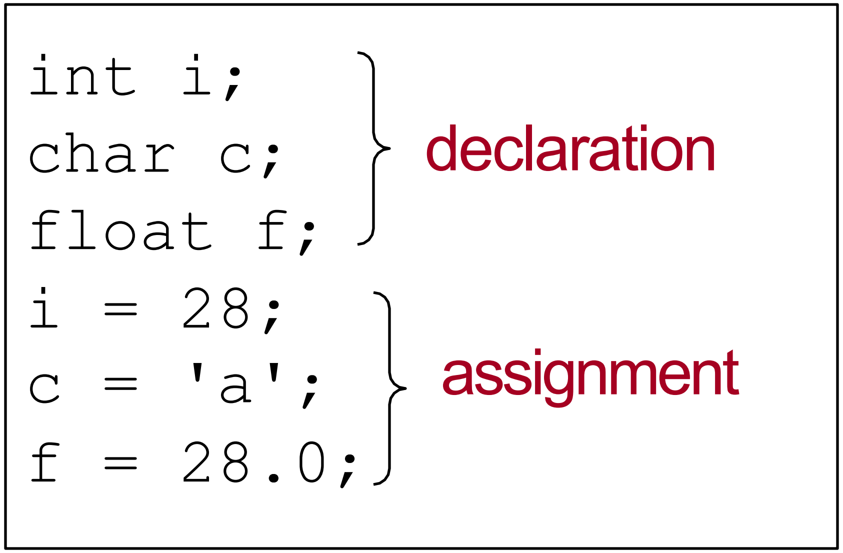
Declaration and Assignment

- Are the following declarations valid?
  - `char c, c;`
  - `char c, C;`
  - `int i`
  - `unsigned int i; float i;`
  - `unsigned int i; float j;`

Declaration and Assignment

- The variables can be assigned values using the assignment operator `='
  - Format
    - `variable_name = value;`
  - E.g.,
    - `i = 10;`
    - `f = 1.2;`
    - `area = 6.28 ;`
    - `area = f;`
- All the variables must be initialized at least once (assigned a value) before their values are used; otherwise, there will be a warning.

Declaration **Before** Assignment



```
int i;
char c;
float f;
i = 28;
c = 'a';
f = 28.0;
```

declaration

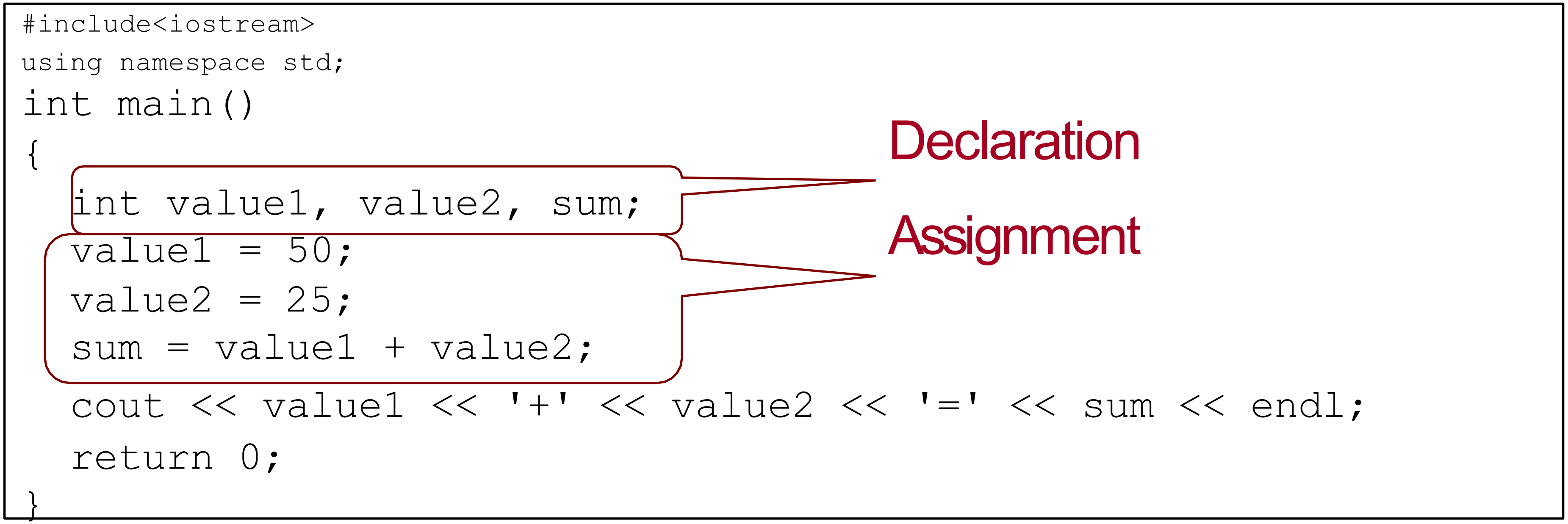
assignment

Declaration **Before** Assignment

```
int i = 28;
char c = 'a';
float f = 28.1;
```

declaration
and
assignment

Declaration **Before** Assignment



```
#include<iostream>
using namespace std;
int main()
{
   int value1, value2, sum;
   value1 = 50;
   value2 = 25;
   sum = value1 + value2;
   cout << value1 << '+' << value2 << '=' << sum << endl;
   return 0;
}
```

Declaration

Assignment

Sum.cpp

Declaration and Assignment

```
int i1;
char 2c
float f;

i1 = 28.5;
2c = '*';
f = 28;
```

Any problems??

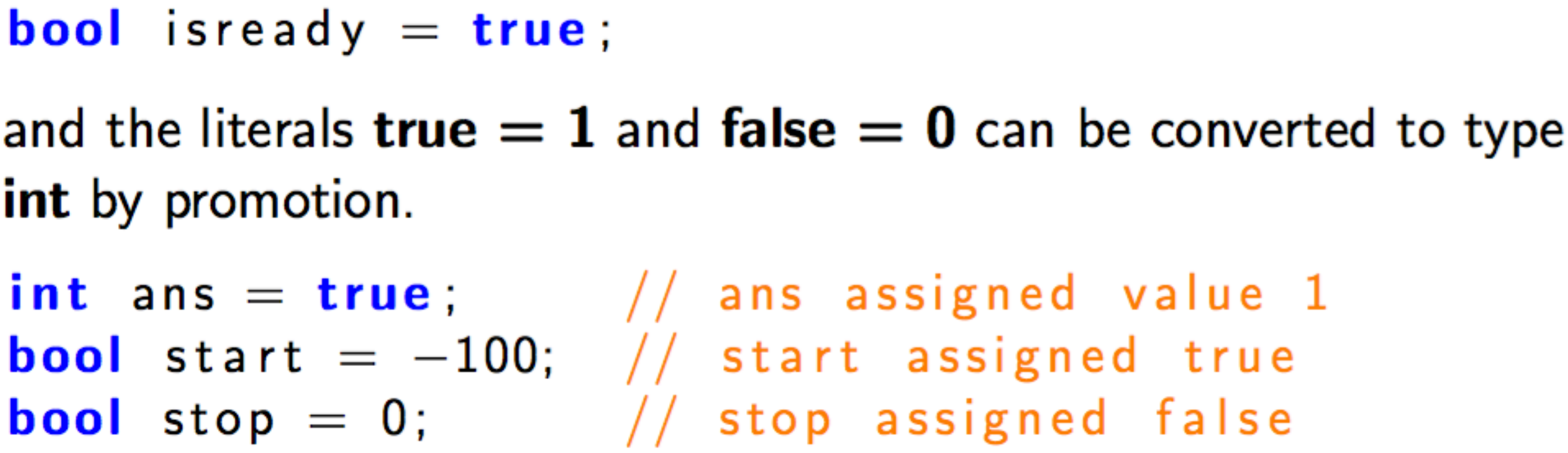
Declaration and Assignment

```
int i;
char c;
float f;

i = 28;
c = 65;
f = 28;
```

Is this OK??

Assignment to Boolean Variable



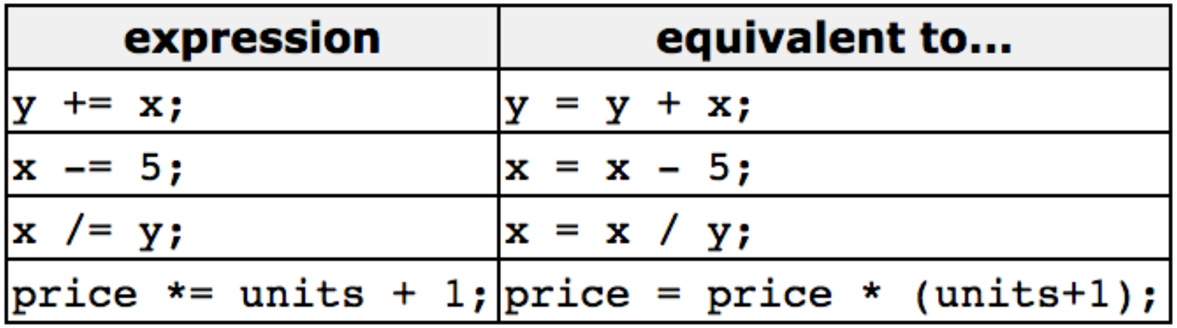
```
bool isready = true;
```

and the literals **true = 1** and **false = 0** can be converted to type **int** by promotion.

```
int ans = true;        // ans assigned value 1
bool start = -100;     // start assigned true
bool stop = 0;         // stop assigned false
```

Compound assignment

- **+=, -=, *=, /=, %=, >>=, <<=, &=, ^=, |=**



expression	equivalent to...
`y += x;`	`y = y + x;`
`x -= 5;`	`x = x - 5;`
`x /= y;`	`x = x / y;`
`price *= units + 1;`	`price = price * (units+1);`

Examples

- Example 1

```
x=(y=3)+1;      /*1. y is assigned 3 */
               /* 2. the value of (y = 3) is 3 */
               /* 3. x is assigned 4 */
```

- Example 2

```
y=3;        /* y is assigned 3 */
x+=y+1;     /* x=x+(y+1) */
```

Class Exercises

- Can you explain these expressions?
  - `x=(y=5)+3;`
  - `x=y=5+3;`
  - `x==(y=5);`

Type Conversion

Type Conversion

- C++ allows for conversions between the basic types, implicitly or explicitly.
- Explicit conversion uses the cast operator.

```
int x = 10;
float y = 3.14,z = 3.14;
y = (float)x;    /* y = 10.0 */
x = (int)z;      /* x = 3     */
x = (int)(-z);   /* x = -3 - rounded approaching zero */
y = x;          /* y = ??? */
```

cast operator

Implicit Conversion

- If the compiler expects one type at a position, but another type is provided, then implicit conversion occurs.

```
int x = 10;
float y = 3.14,z = 3.14;
y =(float)x;   /* y = 10.0 */
x =(int)z;     /* x = 3     */
x =(int)(-z); /* x = -3 - rounded approaching zero */
y = x;           /* y = -3.0 */
```

Implicit conversion

Implicit Conversion

- If the compiler expects one type at a position, but another type is provided, then implicit conversion occurs.

```
int x = 10;
float y = 3.14,z = 3.14;
y =(float)x;   /* y = 10.0 */
x =(int)z;     /* x = 3     */
x =(int)(-z);  /* x = -3 - rounded approaching zero */
x = y;           /* x = ??? */
```

Implicit conversion → possible loss of data !

Implicit Conversion

- If the compiler expects one type at a position, but another type is provided, then implicit conversionoccurs.

```
char c = 'a';
int i;
i = c;   /* i'value is the ASCII code of 'a' */
```

Type: `char`

Type: `int`

Class Exercises

Compare the results of y

```
int i = 5;
float j = 11;
Float y;
y = j / i;
```

```
int i = 5, j = 11;
float y;
y = j / i;
```

```
int i = 5, j = 11;
float y;
y = (float)j / i;
```

Class Exercises

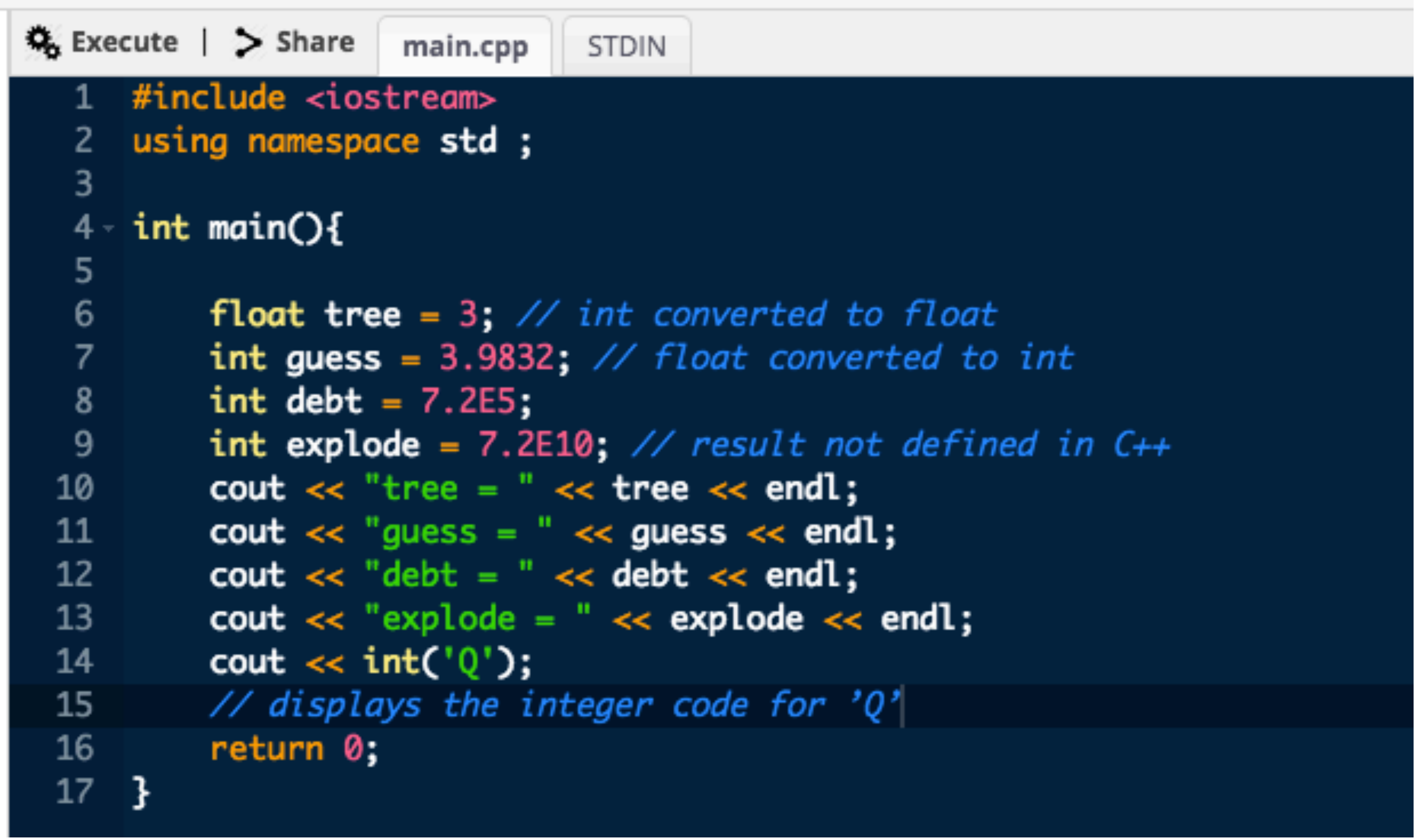
Compare the results of y

```
int i = 5;
float j = 11;
float y;
y = j / i;           /* y = 11.0/5 = 2.2  */
```

```
int i = 5, j = 11;
float y;
y = j / i;           /* y = 11/5 = 2 */
```

```
int i = 5, j = 11;
float y;
y = (float)j / i;    /* y = 11.0/5 = 2.2 */
```

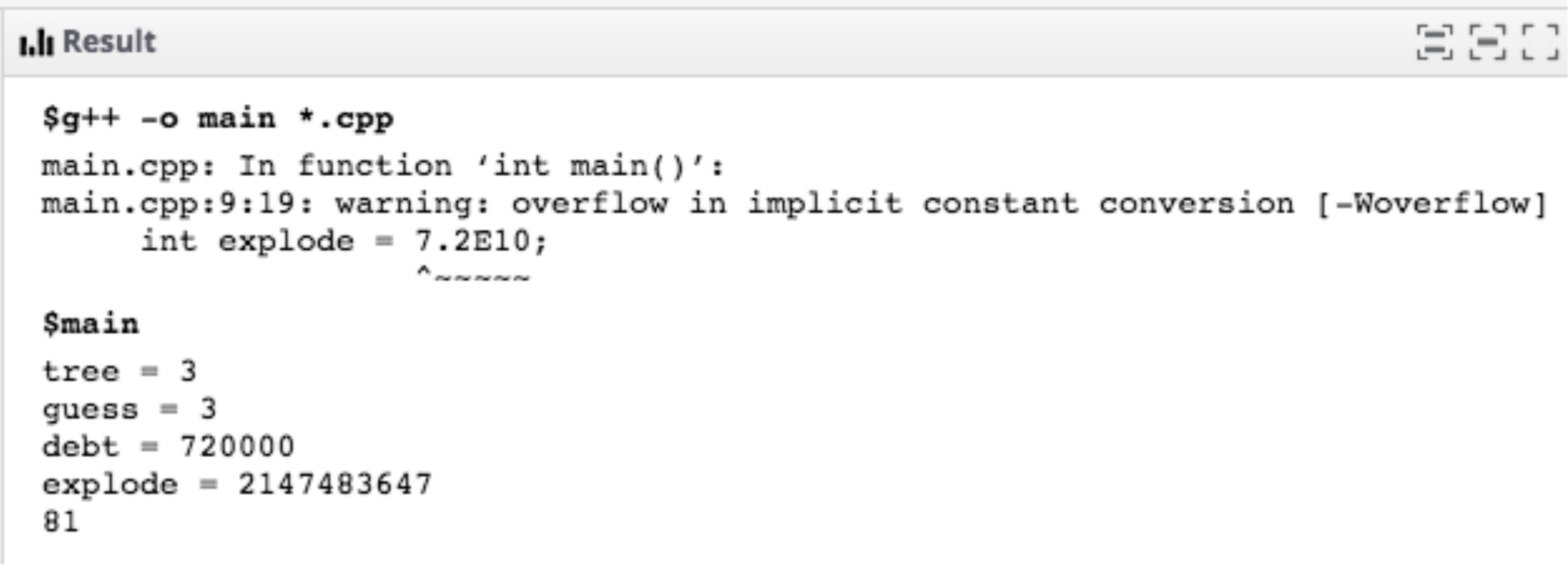
Class Exercises



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

int main(){

    float tree = 3; // int converted to float
    int guess = 3.9832; // float converted to int
    int debt = 7.2E5;
    int explode = 7.2E10; // result not defined in C++
    cout << "tree = " << tree << endl;
    cout << "guess = " << guess << endl;
    cout << "debt = " << debt << endl;
    cout << "explode = " << explode << endl;
    cout << int('Q');
    // displays the integer code for ’Q’
    return 0;
}
```



```
$g++ -o main *.cpp
main.cpp: In function ‘int main()’:
main.cpp:9:19: warning: overflow in implicit constant conversion [-Woverflow]
     int explode = 7.2E10;
                   ^~~~~~
$main
tree = 3
guess = 3
debt = 720000
explode = 2147483647
81
```