AMA2222 Principles of Programming

Leung Man Kin, Adam
Instructor

adam.leung@polyu.edu.hk
TU720

Chapter 1: input and output, variables, arithmetic, file I/O, console output formatting

Mode of teaching:

Lecture: Face-to-face, with recording

Lab: Face-to-face, no recording

Time and venue:

Lecture: Mon 14:30-16:30 (M301) Lab: Thu 16:30-17:30 (M301)

Consultation hours: Mon 16:30-17:30, Thu 15:30-16:30 (TU720)

* 10/04/2023 (Mon) is a public holiday. Lecture will be cancelled.

Assessment method:

Continuous Assessment	Quizzes	12% (2 quizzes)
	Lab project	24% (10 lab projects)
	Midterm Test	24%
Final Examination		40%

Week	Lecture (mon)	Lab (thu)
1	lecture 1	lab 1
2	lecture 2	lab 2
Lunar New Year Break		
3	lecture 3	lab 3
4	lecture 4	lab 4
5	lecture 5	quiz 1
6	lecture 6	lab 5
7	lecture 7	lab 6
8	lecture 8	lab 7
9	**midterm**	lab 8
10	lecture 9	lab 9
11	lecture 10	quiz 2
12	lecture 11	lab 10
13	(holiday)	lecture 12

Reference books

Y. Daniel Liang, Introduction to Programming with C++, 3rd edition, Pearson, 2014

Thomas H. Cormen et al, Introduction to algorithms, 3rd edition, MIT Press, 2009

Development platforms for writing C++ programs:

(installation needed, recommended) https://www.bloodshed.net/

(installation needed) http://orwelldevcpp.blogspot.com/

(installation not needed, work online) http://cpp.sh/

Syllabus:

1) input and output, variables, arithmetic, file I/O, console output formatting	1L
2) math expressions, logical operation, if else statement, switch statement	1L
3) control statement, for loop, while loop	1L
4) functional programming, function, local and global variable, lambda function	1L
5) data structures, character and strings, array	2L
6) algorithm I, swapping, searching and sorting,	1L
7) algorithm II, divide and conquer, greedy, recursion	1L
8) object oriented programming, class and object, inheritance	2L
9) random simulation, numerical methods, applications	1L
10) pointer and dynamic memory	1L

What is a computer?

A computer is an electronic device that stores and process data.

What is programming?

Programming is the process of writing computer programs, which are instructions that tell a computer what to do. We can use the IPO model to describe the work of a computer program.

input -> process -> output

Three levels of programming languages:

Machine language

binary code for built-in primitive instructions

Assembly language

coding with short descriptive words to represent machine language instructions

High-level language **C++ is high-level language**

platform-independent, easy to use and learn, need to be translated into machine-code by a compiler

Example of a Python program (Python3): **Just one line!**

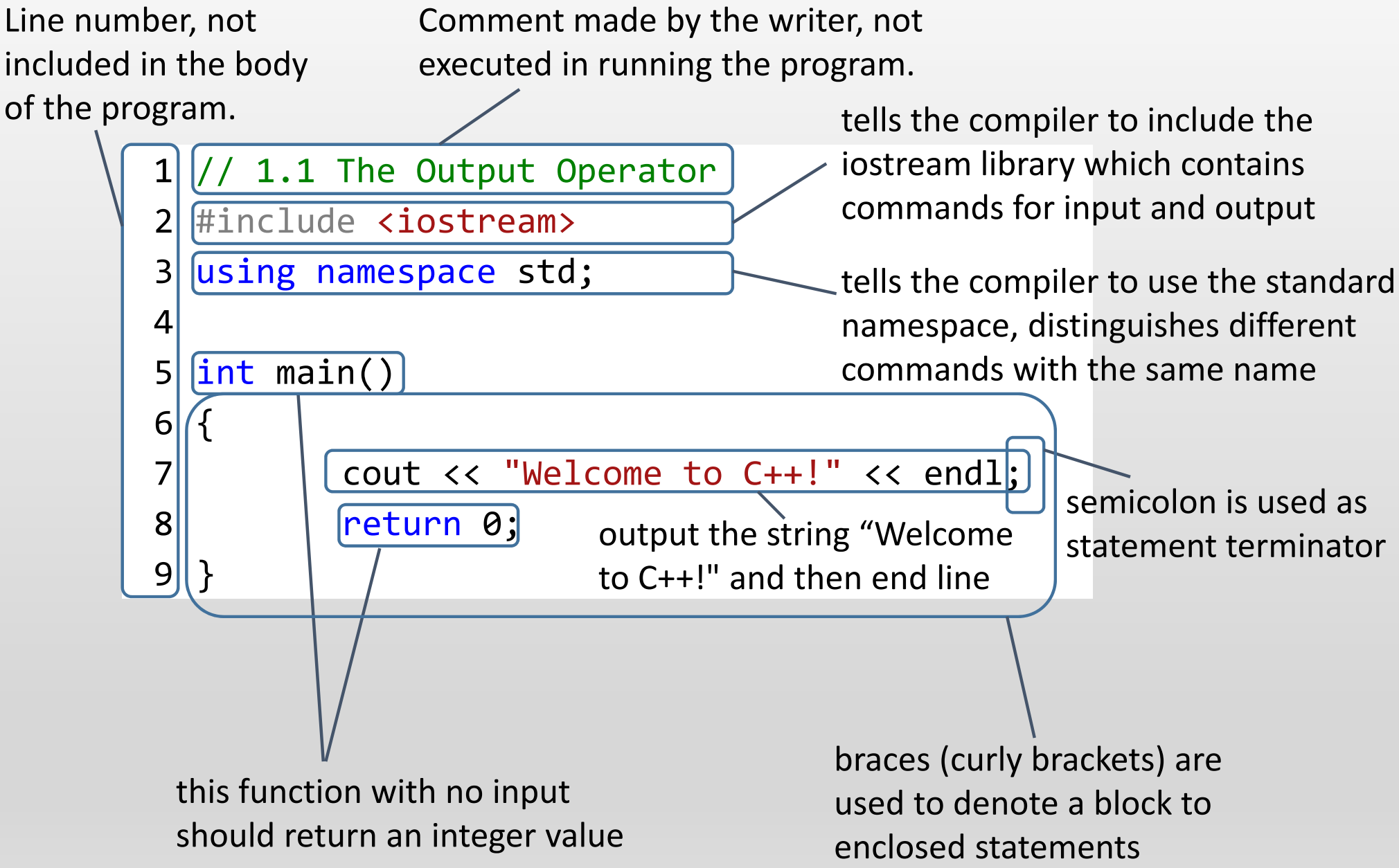
```
In [1]: print("Welcome to python!")
```

Example of a C++ program:

Example program 1.1 The Output Operator

```
1 // 1.1 The Output Operator
2 #include <iostream>
3 using namespace std;
4
5 int main()
6 {
7         cout << "Welcome to C++!" << endl;
8         return 0;
9 }
```

Normally a C++ program has more lines compared to Python, but it is good for beginner as it introduce more basic concepts.


Line number, not included in the body of the program.
Comment made by the writer, not executed in running the program.
tells the compiler to include the iostream library which contains commands for input and output
tells the compiler to use the standard namespace, distinguishes different commands with the same name
semicolon is used as statement terminator
output the string “Welcome to C++!" and then end line
this function with no input should return an integer value
braces (curly brackets) are used to denote a block to enclosed statements
1 // 1.1 The Output Operator
2 #include <iostream>
3 using namespace std;
4
5 int main()
6 {
7 cout << "Welcome to C++!" << endl;
8 return 0;
9 }

What is a variable?

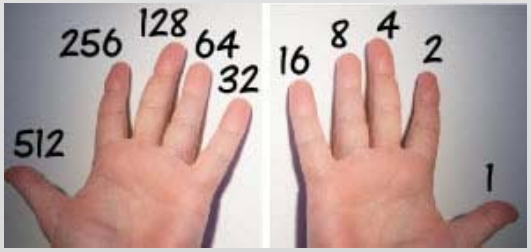
A **variable** is a **value that can change** depending on the condition and information passed to the program. All variables must be declared with a **data type** and a **name** before they can be used in a program, eg `int x;` declares an integer x.

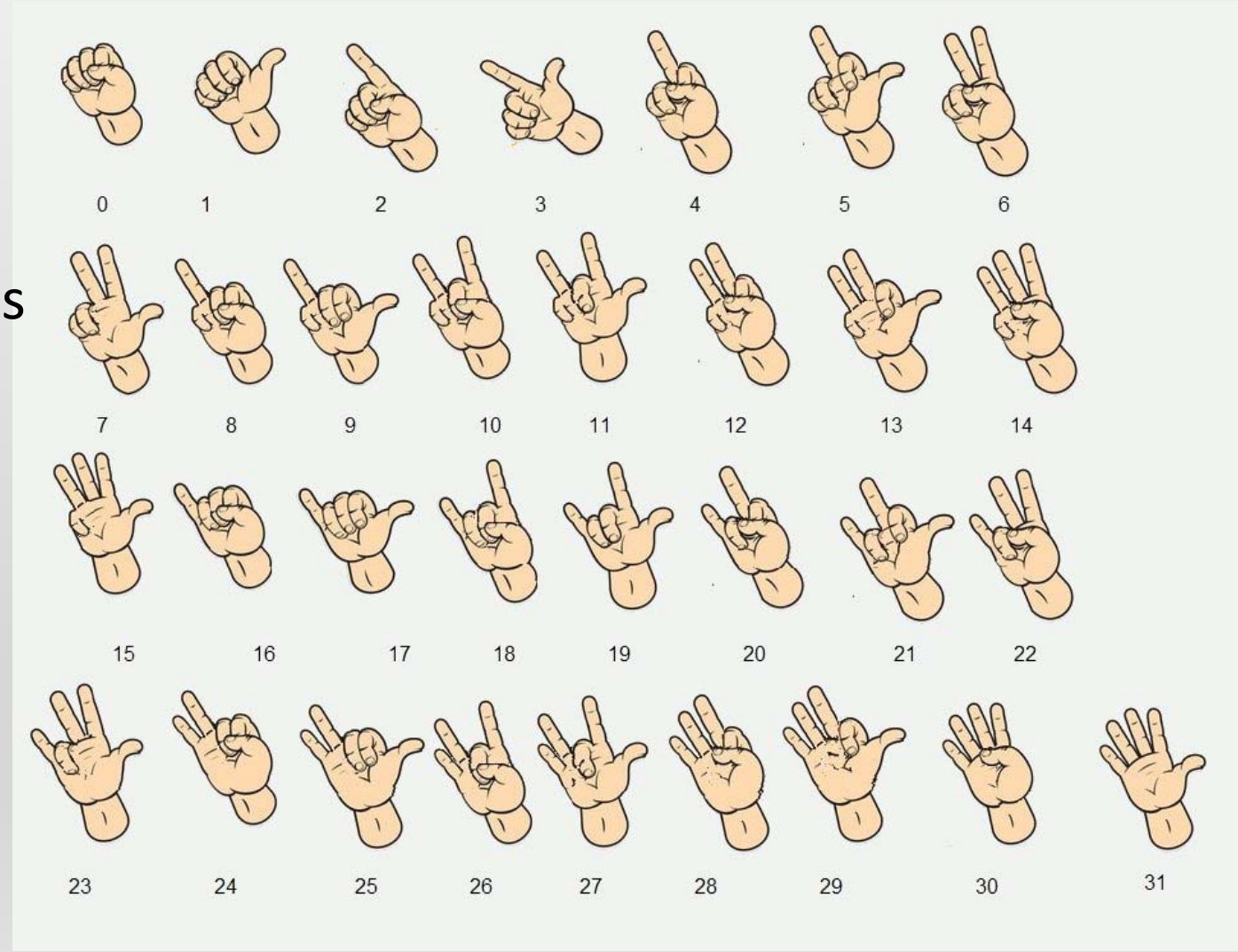
In choosing a suitable data type for the variable, please be aware of the range and also the storage space. One bit is the smallest unit of data stored in a computer, which can be 0 or 1 in binary form. So if a data is stored as n bits, it can take 2^n possible values. A byte is a unit with 8 bits for practical purpose.

Data type	Range	Bit size	Byte size
bool	0 to 1 *8 bits, but the top 7 bits are ignored	1*	1
char	0 to 255	8	1
int	-2^{31} to $2^{31}-1$	32	4
float	about 3.4×10^{-38} to 3.4×10^{38}	32	4
double	about 1.7×10^{-308} to 1.7×10^{308}	64	8

In kindergarten, we learnt counting 1 to 10 by your ten fingers. But actually, what is the bit size and greatest possible number of values stored by your fingers?

ans:
10 bits
1024 possible values





How to assign a value into a variable?

(1) Assign an initial value when declaring: `int x = 3;`

(2) Direct input from keyboard: `cin >> x;`

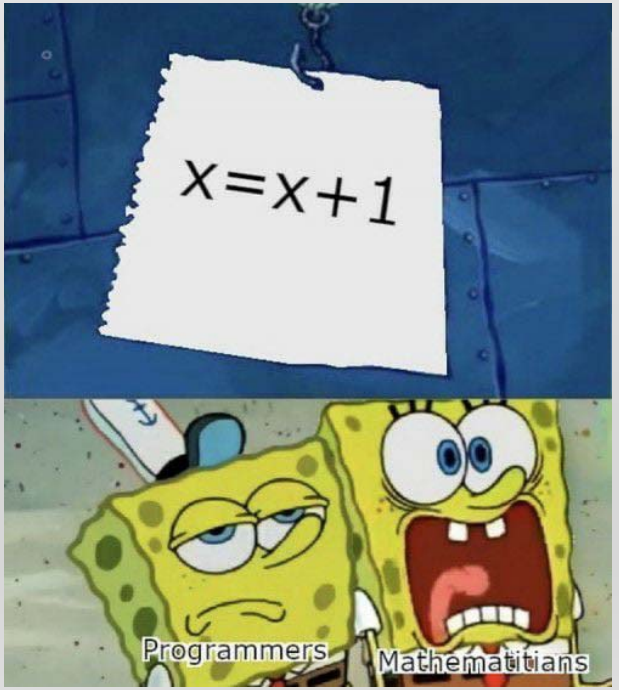
(3) Assigned within the program using `=` , e.g.:

```
x = 1;
```

```
x = 2 + 3 * 4;
```

```
x = y;
```

```
x = x + 1;
```



Input and output

Input and output in C++ can be performed by the `cin` and `cout` commands, followed by arrows of opposite directions `>>` and `<<` respectively. Refer to the syntax below:

To input some value(s) and assign to some variable(s)

```
cin >> (variable);
cin >> (variable1) >> (variable2) >> (variable3);
```

To display a value on the screen

```
cout << (value);
```

To display the value(s) of some variable(s) on screen

```
cout << (variable);
cout << (variable1) << (variable2) << (variable3);
```

Classwork 1.1

Evaluate the following coding. Find the outputs.

(You may use `cout << endl;` to separate the outputs.)

```
int x1;
cout << x1;
```

```
int x2;
x2 = 2.5;
cout << x2;
```

```
int x3;
x3 = 90000000000;
cout << x3;
```

```
int x4;
x4 = 1/0;
cout << x4;
```

Notes to the classwork:

- the default value for a numerical variable is 0 commonly, however it might be machine dependent
- for an integer variable, if we assign a float value such as 2.5 it will be truncated to an integer
- if a value out of the range is assigned to a variable, overflow error occurs
- if we assign an undefined value like 1/0, error occurs

Additional notes:

- if we assign x with a value dependent on other variables, make sure that those variables have been declared
- the equal sign means “is assigned with” rather than our usual mathematical meaning “equals to”

Example program 1.2

```
1  //1.2 Variables, Arithmetic, and Input Operator
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7          int number1, number2, sum;
8
9          cout << "Enter the first integer: ";
10         cin >> number1;
11
12         cout << "Enter the second integer: ";
13         cin >> number2;
14
15         sum = number1 + number2;
16
17         cout << "The sum is " << sum << endl;
18
19         return 0;
20 }
```

Example program 1.2

```
1  //1.2 Variables, Arithmetic, and Input Operator
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7          int number1, number2, sum;
8
9          cout << "Enter the first integer: ";
10         cin >> number1;
11
12         cout << "Enter the second integer: ";
13         cin >> number2;
14
15         sum = number1 + number2;
16
17         cout << "The sum is " << sum << endl;
18
19         return 0;
20 }
```

declaring the variables by stating the data type (integer) and then the name.

remind the user what to input

value input to number1 from keyboard

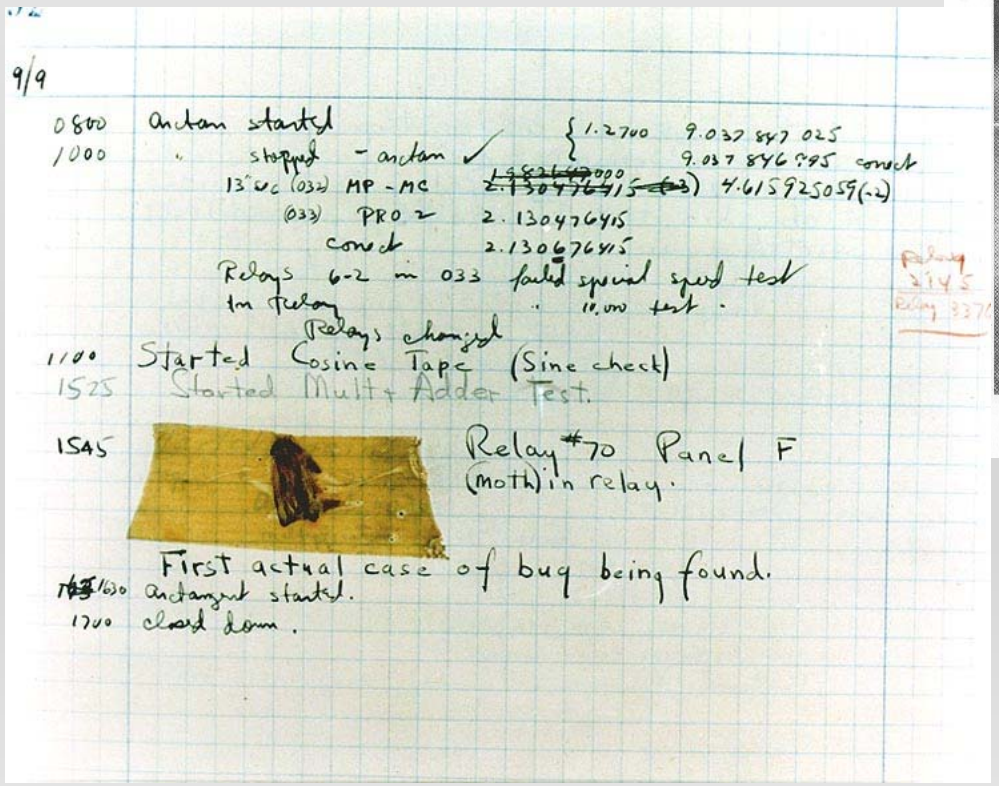
value input to number2 from keyboard

value assigned to sum by arithmetic operation

display the result

Debugging

A "bug" refers to an error or flaw that result in an runtime error or incorrect result of a program. The process of finding and fixing bugs is called debugging.



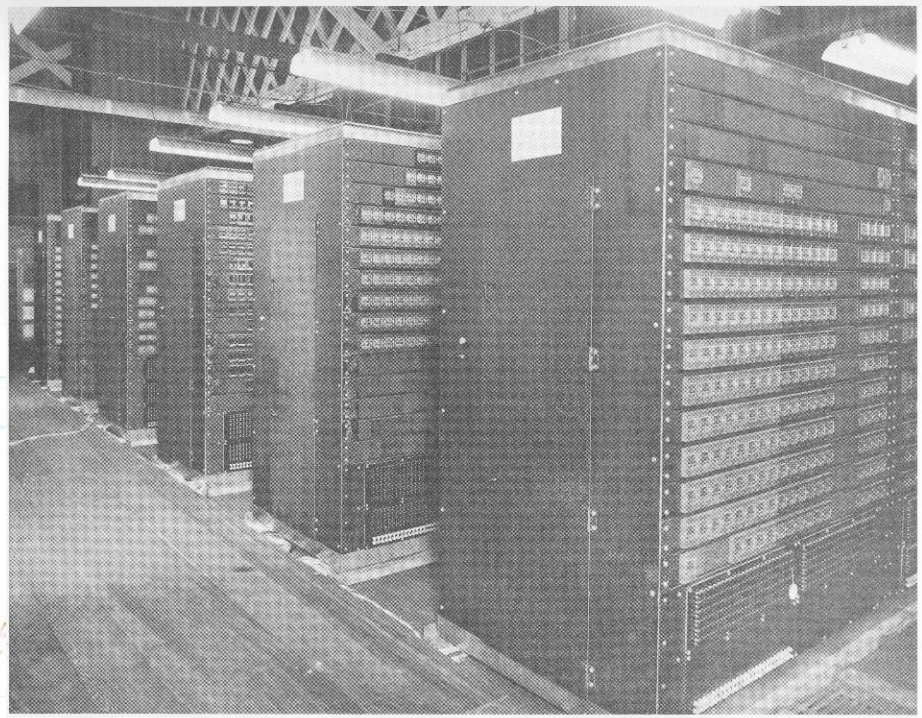


Figure 5 Mark II: Relay Cubicles

A page from the Harvard Mark II electromechanical computer's log, featuring a dead moth that was removed from the device. (1947)

Classwork 1.2:

Find the bugs in the following codes:

```
#include <iostream>
using namespace std;
int main()
{
        cout >> "Hello" >> endl
        return 0;
}
```

```
#include <iostream>
using namespace std;
int main()
{
        int x;
        cin >> "x" ;
        x = x + 1;
        cout << "next is " << "x";
        return 0;
}
```

Solution to classwork 1.2:

Find the bugs in the following codes:

```
#include <iostream>
using namespace std;
int main()
{
        cout >> "Hello" >> endl
        return 0;
}
```

cout should be followed by << instead of >>

there should be a ; after endl

```
#include <iostream>
using namespace std;
int main()
{
        int x;
        cin >> "x" ;
        x = x + 1;
        cout << "next is " << "x";
        return 0;
}
```

"x" is a string but not the integer variable x

Arithmetic operations in C++

Operation	Operator	Algebraic Expression	C++ Expression
Addition	+	$f + 7$	`f + 7`
Subtraction	-	$p - c$	`p - c`
Multiplication	*	$b \times m$	`b * m`
Division	/	$x \div y$	`x / y`
Modulus	%	$r \bmod s$	`r % s`

Operator	Operation	Precedence
`( )`	Parentheses	First. If the parentheses are nested, the innermost pair is evaluated first. The order of evaluation of two sets of parentheses that are not nested, however, is not specified in the C++ standard.
`* / %`	Multiplication, Division, Modulus	Second. If there are several, they're evaluated from left to right.
`+ -`	Addition, Subtraction	Third. If there are several, they're evaluated from left to right.

Classwork 1.3

Notice: result of arithmetic operation depends on the data type.

Classwork exercise: evaluate the output of the following.

```
int x;
int y;
x = 4;
y = 3;
cout << x/y << endl;
```

1

```
int x;
double y;
x = 4;
y = 3;
cout << x/y << endl;
```

1.33333

```
double x;
int y;
x = 4;
y = 3;
cout << x/y << endl;
```

1.33333

```
double x;
double y;
x = 4;
y = 3;
cout << x/y << endl;
```

1.33333

Named constant

Once defined initially, a constant will not be changed upon the execution of a program. You can also define your own constant in the program, eg

```
const double inflation = 1.06;
```

defines an inflation of 6% used throughout the program. If you want to adjust the inflation rate to 5%, you only need to change this constant value rather than searching and replacing in the whole program.

Example program 1.3

```
 1 // Section 1.3 Named Constant
 2 #include <iostream>
 3 using namespace std;
 4 int main()
 5 {
 6         const double PI = 3.14159;
 7
 8         double radius;
 9         cout << "Enter a radius: ";
10         cin >> radius;
11
12         double area = radius * radius * PI;
13
14         cout << "The area is ";
15         cout << area << endl;
16
17         return 0;
18 }
```

Example program 1.3

```
1  // Section 1.3 Named Constant
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6          const double PI = 3.14159;
7
8          double radius;
9          cout << "Enter a radius: ";
10         cin >> radius;
11
12         double area = radius * radius * PI;
13
14         cout << "The area is ";
15         cout << area << endl;
16
17         return 0;
18 }
```

define the constant (pi) we need to use

step 1: read the radius

step 2: compute the area

step 3: display the area

Classwork 1.3:

Write a program that inputs three integers from the keyboard and prints the sum, average, and product of these numbers. The screen dialog should appear as follows:

Please enter three integers: **12 27 14**
The sum is 53
The average is 17.6667
The product is 4536

Solution to classwork 1.3

```
 1 #include <iostream>
 2 using namespace std;
 3 int main()
 4 {
 5         int n1, n2, n3;
 6         cout << "Please enter three integers:";
 7         cin >> n1 >> n2 >> n3;
 8         cout << "The sum is " << n1 + n2 + n3 << endl;
 9         cout << "The average is " << (n1 + n2 + n3)/3.0 << endl;
10         cout << "The product " << n1 * n2 * n3 << endl;
11         return 0;
12 }
```

Notice that (n1+n2+n3)/3 will fail as it is regarded as integer division. An alternative solution is to convert the values from integer to double first before taking division.

```
 9         cout << "The average is " << (double)(n1+n2+n3)/3 << endl;
```

Augmented operators

The operators +, -, *, /, and % can be combined with the assignment operator to form **augmented operators**:

Operator	Example	Equivalent
+=	i += 8	i = i + 8
-=	i -= 8	i = i - 8
*=	i *= 8	i = i * 8
/=	i /= 8	i = i / 8
%=	i %= 8	i = i % 8

The augmented assignment operator is performed last after the other operators in the expression are evaluated.

Increment and decrement operators

The **increment operator** and **decrement operator** are for incrementing and decrementing a variable by 1:

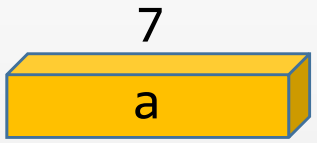
Operator	Name	Example (assume i=1)
++var	preincrement	`int j= ++i; // j=2, i=2`
var++	postincrement	`int j= i++; // j=1, i=2`
--var	predecrement	`int j= --i; // j=0, i=0`
var--	postdecrement	`int j= i--; // j=1, i=0`

Classwork 1.4:

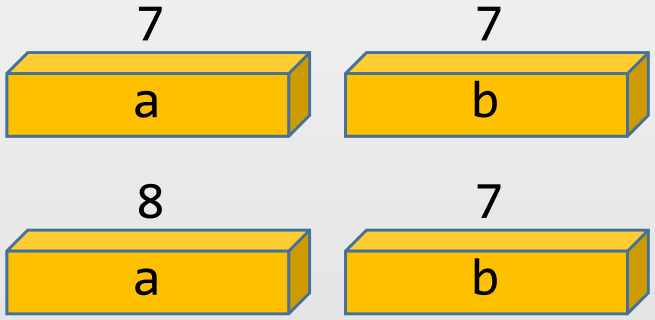
Evaluate the output of the following programs:

```
#include <iostream>
using namespace std;
int main()
{
        int a = 7;
        int b = a++;
        int c = --b;
        int d = c--;
        cout << "a is " << a << endl;
        cout << "b is " << b << endl;
        cout << "c is " << c << endl;
        cout << "d is " << d << endl;
        return 0;
}
```

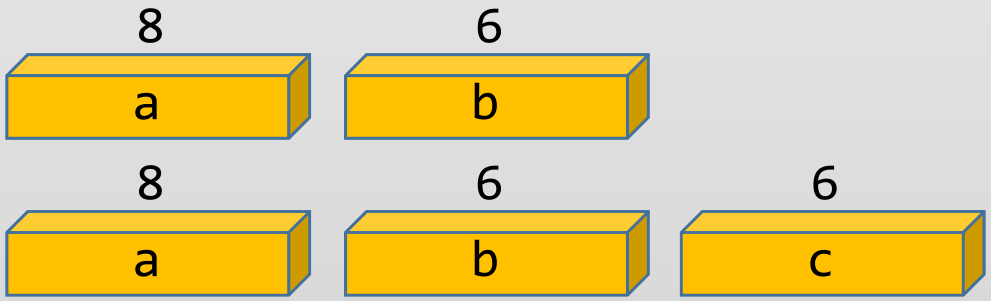
```
int a = 7;
```



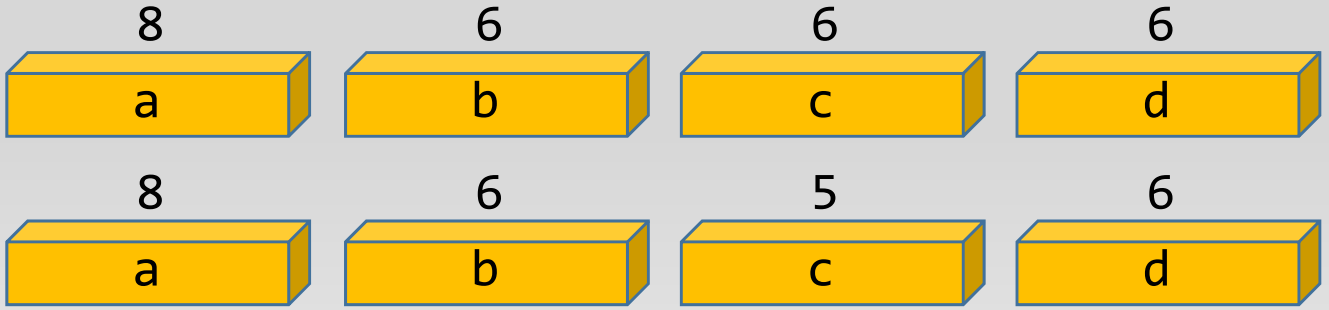
```
int b = a++;
```



```
int c = --b;
```



```
int d = c--;
```



File input and output

We have been using direct input from keyboard and output to the screen. However, for practical purpose, we need to input or output with a text file. With a file declared and opened, we can use `<<` and `>>` notation for output and input respectively.

The functions and keywords related to file input/output are put in the header file `<fstream>`.

Two types of variables should be declared before:

`ofstream` is used for declaring output files.

`ifstream` is used for declaring input files.

Two functions for file opening and closing:

`.open("  ")` assigns the actual file name (in "") to the file

`.closed` closes the file

Example program 1.4 Outputting to a text file.

```
 1 #include <iostream>
 2 #include <fstream>
 3 using namespace std;
 4 int main()
 5 {
 6           ofstream doc1;
 7           doc1.open("test_output_file.txt");
 8           doc1 << 123 << endl;
 9           doc1 << "hi auntie!";
10           doc1.close();
11           return 0;
12 }
```

The output file is created as test_output_file.txt

```
123
hi auntie!
```

Example program 1.5 Inputting data from a text file.

A text file containing the name and GPA of a student is stored in a text file student_gpa.txt

```
Sze Ka Ka
3.69
```

```
 1 #include <iostream>
 2 #include <fstream>
 3 #include <string>
 4 using namespace std;
 5 int main()
 6 {
 7           ifstream file;
 8           file.open("student_gpa.txt");
 9           string name;
10           double gpa;
11           getline(file,name);
12           file >> gpa;
13           cout << name << " has a GPA of " << gpa;
14           file.close();
15           return 0;
16 }
```

Formatting console output

C++ provides additional functions for formatting how a value is displayed. These functions are called stream manipulators and are included in the `<iomanip>` header.

Function	Description	Effect
`setprecision(n)`	sets the precision of a `float` or `double` number	Permanent
`fixed`	displays a `float` or `double` numbers in nonscientific (fixed) notation. By default, the fixed number of digits after the decimal point is 6.	Permanent
`scientific`	displays floating-point numbers in scientific notation	Permanent
`showpoint`	causes a `float` or `double` numbers to be displayed with a decimal point with trailing zeros even if it has no fractional part.	Permanent
`setw(width)`	specifies the width of a print field	Temporary
`left`	justifies the output to the left	Permanent
`right`	justifies the output to the right (default)	Permanent

Example program 1.6a

```
 1 #include <iostream>
 2 #include <iomanip>
 3 using namespace std;
 4 int main()
 5 {
 6           double amount = 12618.98;
 7           double interestRate = 0.0013;
 8           double interest = amount * interestRate;
 9           cout << "Interest is " << interest << endl;
10           cout << "Interest is " << fixed << setprecision(2)
11                     << interest << endl;
12           return 0;
13 }
```

Result:

```
Interest is 16.4047
Interest is 16.40
```

The actual answer is 16.404674. By default, 6 significant figures will be shown. By using `fixed` and then `setprecision` to be 2, only 2 decimal places will be shown.

Example program 1.6b

```
 1 #include <iostream>
 2 #include <iomanip>
 3 using namespace std;
 4 int main()
 5 {
 6           double number = 12.34567;
 7           cout << setprecision(3) << number << " "
 8                     << setprecision(4) << number << " "
 9                     << setprecision(5) << number << " "
10                     << setprecision(6) << number << endl;
11           return 0;
12 }
```

Result:

```
12.3 12.35 12.346 12.3457
```

By using `setprecision` to n,
n significant figures will be shown.

Example program 1.6c

```
 1 #include <iostream>
 2 #include <iomanip>
 3 using namespace std;
 4 int main()
 5 {
 6           cout << setprecision(6);
 7           cout << 1.23 << endl;
 8           cout << showpoint << 1.23 << endl;
 9           cout << fixed << showpoint << 1.23 << endl;
10           return 0;
11 }
```

Result:

```
1.23
1.23000
1.230000
```

We have first `setprecision` to be 6. Since the value to show is exactly 1.23, the zeros at back will not be displayed.

Using `showpoint` , 6 significant digits including zeros will be displayed.

Using `fixed` and `showpoint` , 6 decimal places including zeros will be displayed.

Example program 1.6d

```
 1 #include <iostream>
 2 #include <iomanip>
 3 using namespace std;
 4 int main()
 5 {
 6           cout << right;
 7           cout << setw(8) << 1.23 << endl;
 8           cout << setw(8) << "Hi!" << endl;
 9           cout << left;
10           cout << setw(8) << 1.23 << endl;
11           cout << setw(8) << "Hi!" << endl;
12           return 0;
13 }
```

Result:

```
    1.23
     Hi!
1.23
Hi!
```

Using `setw` we can set the display width to be 8 in case the number or string to display is less than 8 characters. Additional spacing will be displayed.

We can use either `right` or `left` to set the alignment.

Classwork exercise 1.5

Show the output of the following, use ☐ for a space.

```
double monthlyPayment = 1345.4567;
double totalPayment = 866.887234;
cout << setprecision(7);
cout << monthlyPayment << endl;
cout << totalPayment << endl;
cout << fixed << setprecision(2);
cout << setw(8) << monthlyPayment << endl;
cout << setw(8) << totalPayment << endl;
```

Answer to classwork exercise 1.5

Show the output of the following, use ☐ for a space.

`1345.457`	(7 sig. fig.)
`866.8872`	(7 sig. fig.)
☐`1345.46`	(2d.p., width = 8)
☐☐`866.89`	(2d.p., width = 8)