AMA2222 Principles of Programming

Leung Man Kin, Adam
Instructor

adam.leung@polyu.edu.hk
TU720

Chapter 1: Elementary Programming
Introduction, Input and output,
variable, arithmetic operation, constant

Teaching mode

- 2 hours of Lecture
- 1 hour of Lab

Assessment method

- quiz 12% (4% x 3)
- lab project 24% (2% x 12)
- midterm 24%
- final exam 40%

Reference book

Introduction to Programming with C++ (3rd edition) by Daniel Liang

Software needed

Dev-C++	http://orwelldevcpp.blogspot.com/
C++ Shell	http://cpp.sh/

or other C++ integrated development environments

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

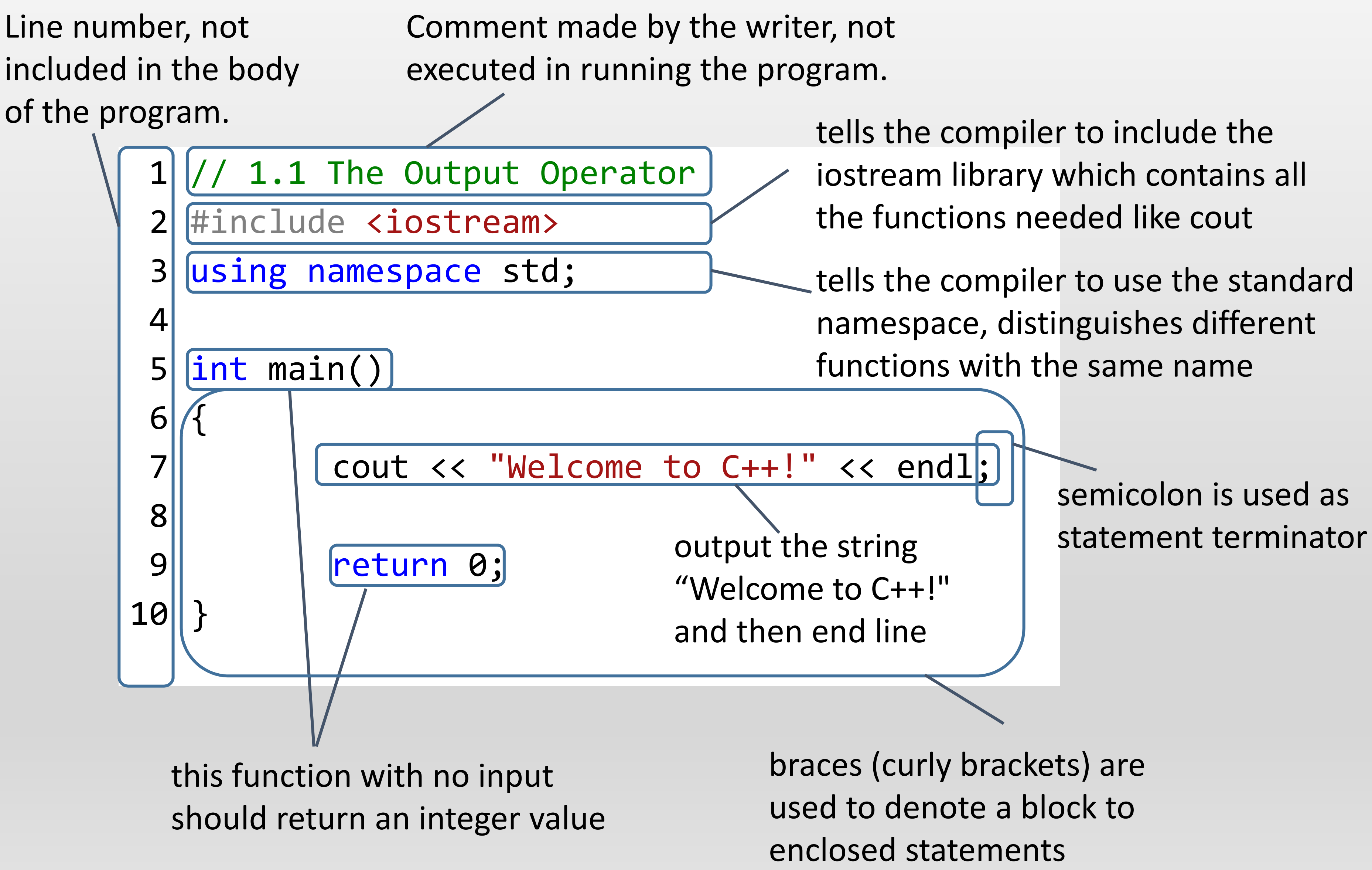
High-level language **C++ is high-level language**

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

Example program 1.1
The Output Operator

```cpp
// 1.1 The Output Operator
#include <iostream>
using namespace std;

int main()
{
	cout << "Welcome to C++!" << endl;
	return 0;
}
```



```
 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
 9        return 0;
10 }
```

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 all the functions needed like cout

tells the compiler to use the standard namespace, distinguishes different functions 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

What is a variable?

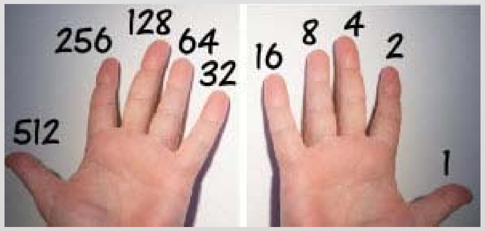
A **variable** is a **value that can change** depending on the condition and information passed to the program. All variables must be declared with a **data type** and a **name** before they can be used in a program, eg `int x;` declares an integer x.

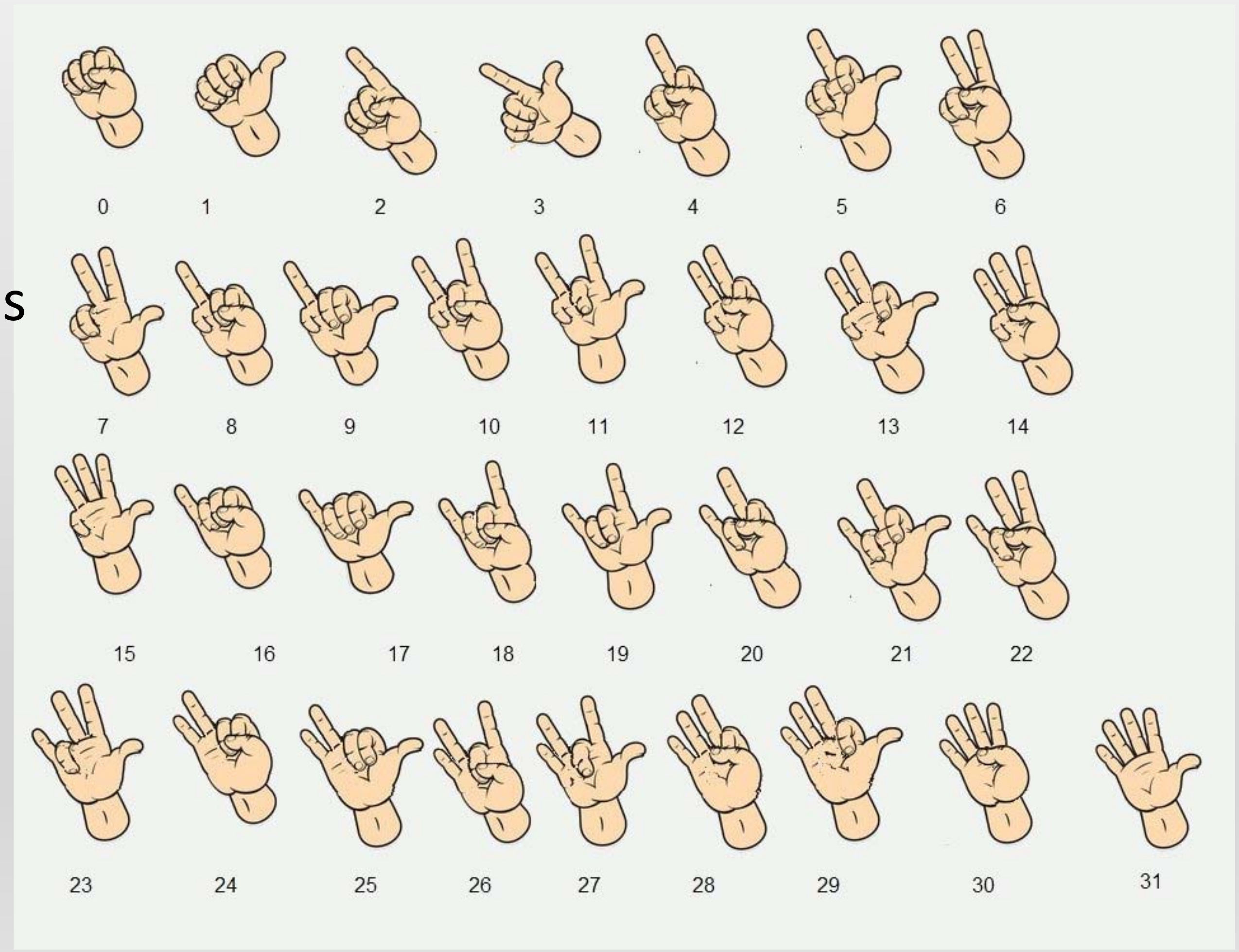
In choosing a suitable data type for the variable, please be aware of the range and also the storage space. One bit is the smallest unit of data stored in a computer, which can be 0 or 1 in binary form. So if a data is stored as n bits, it can take 2^n possible values. A byte is a unit with 8 bits for 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`	$0 \text{ to } 255$	8	1
`int`	$-2^{31} \text{ to } 2^{31} - 1$	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 input a value into a variable?

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

(2) Input from keyboard: `cin >> x;`

(3) Assigned within the program using `=` , eg

```
x = 1;
```

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

```
x = y + z;
```

Notice:

- the default value for an integer variable is 0
- since x is an integer, error occurs if we assign a non-integer value such as 2.5 or undefined value like 3/0
- 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

```cpp
// Section 1.2 Variables, Arithmetics, and Input Operator
#include <iostream>
using namespace std;

int main()
{
        int number1;     // first integer to add
        int number2;     // second integer to add
        int sum;// sum of number1 and number2

        cout << "Enter the first integer: ";
        cin >> number1;

        cout << "Enter the second integer: ";
        cin >> number2;

        sum = number1 + number2;

        cout << "The sum is " << sum << endl;

        return 0;
}
```

Example program 1.2

```cpp
// Section 1.2 Variables, Arithmetics, and Input Operator
#include <iostream>
using namespace std;

int main()
{
        int number1;     // first integer to add
        int number2;     // second integer to add
        int sum;// sum of number1 and number2

        cout << "Enter the first integer: ";
        cin >> number1;

        cout << "Enter the second integer: ";
        cin >> number2;

        sum = number1 + number2;

        cout << "The sum is " << sum << endl;

        return 0;
}
```

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

value input to number1 from keyboard

value input to number2 from keyboard

value assigned to sum by arithmetic operation

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.

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

```cpp
// Section 1.3 Named Constant
#include <iostream>
#include <cmath>
using namespace std;

int main()
{
        const double PI = 3.14159;

        double radius;
        cout << "Enter a radius: ";
        cin >> radius;

        double area = radius * radius * PI;

        cout << "The area is ";
        cout << area << endl;

        return 0;
}
```

Example program 1.3

```cpp
// Section 1.3 Named Constant
#include <iostream>
#include <cmath>
using namespace std;

int main()
{
    const double PI = 3.14159;

    double radius;
    cout << "Enter a radius: ";
    cin >> radius;

    double area = radius * radius * PI;

    cout << "The area is ";
    cout << area << endl;

    return 0;
}
```

define the constant (pi) we need to use

step 1: read the radius

step 2: compute the area

step 3: display the area

Classwork 2:

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 the first integer: 12
Please enter the second integer: 27
Please enter the third integer: 14
The sum is 53
The average is 17.6667
The product is 4536
```

Solution to classwork 2

```cpp
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
	int n1;
	int n2;
	int n3;
	cout << "Please enter the first integer:";
	cin >> n1;
	cout << "Please enter the second integer:";
	cin >> n2;
	cout << "Please enter the third integer:";
	cin >> n3;
	cout << "The sum is " << n1 + n2 + n3 << endl;
	cout << "The average is " << (n1 + n2 + n3) /3.0 << endl;
	cout << "The sum product " << n1 * n2 * n3 << endl;
	return 0;
}
```

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:

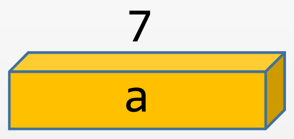
Operator	Name	Example (assume i=1)
++var	preincrement	int j= ++i; // j=2, i=2
var++	postincrement	int j= i++; // j=1, i=2
--var	predecrement	int j= --i; // j=0, i=0
var--	postdecrement	int j= i--; // j=1, i=0

Classwork 2:

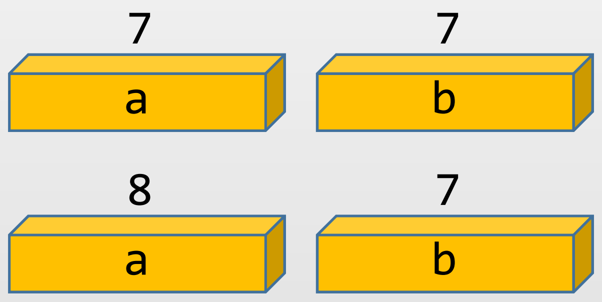
Evaluate the output of the following programs:

```cpp
#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;
}
```

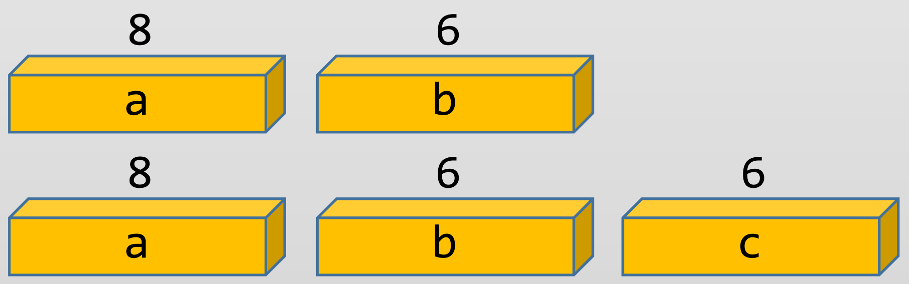
```
int a = 7;
```



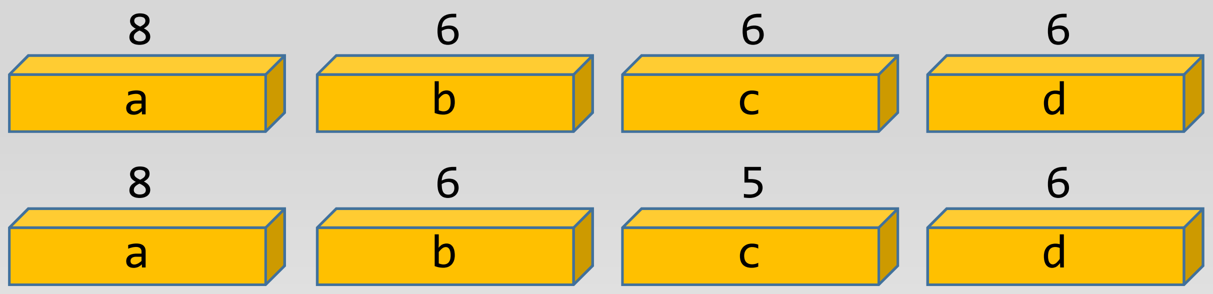
```
int b = a++;
```



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

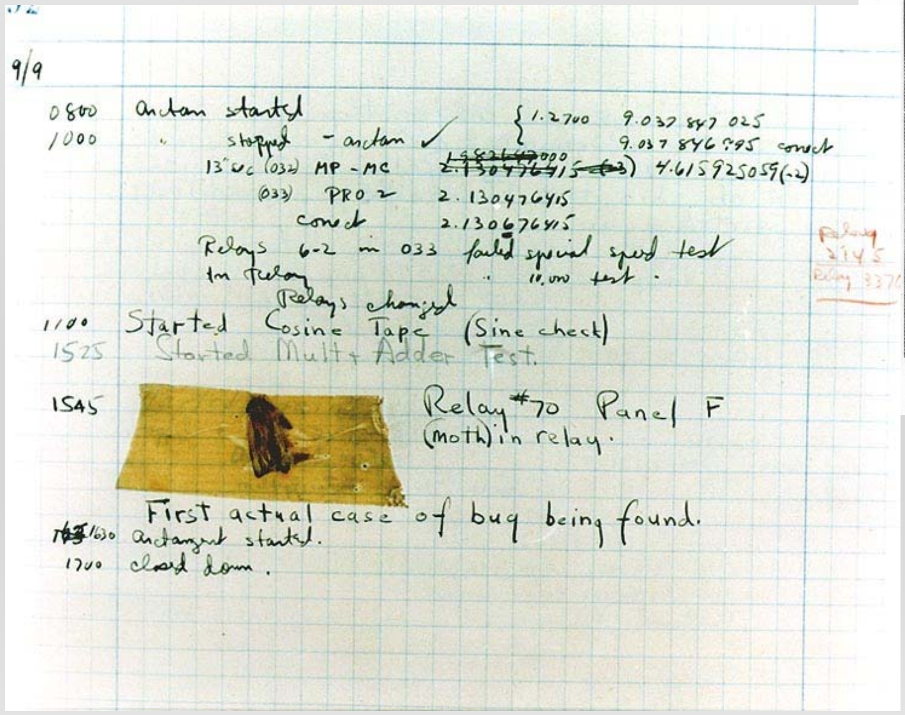


```
int d = c--;
```



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.



```
9/9

0800   Anctan started
1000     "   stopped - anctan ✓          { 1.2700   9.037 847 025
                                          {          9.037 846 995 conect
         13" uc (032) MP - MC    ~~1.982147000~~
                                 ~~2.130476415~~ (~~-3~~) 4.615925059(-2)
               (033) PRO 2         2.130476415
                     conect        2.130676415
         Relays 6-2 in 033 failed special speed test
         In relay        "    11,000 test .
                 Relays changed
1100   Started Cosine Tape (Sine check)
1525   Started Mult+ Adder Test.

1545                         Relay #70 Panel F
                             (moth) in relay.

        First actual case of bug being found.
~~[illegible]~~1630 anctangent started.
1700   closed down.
```

Relay 2145
Relay 3370

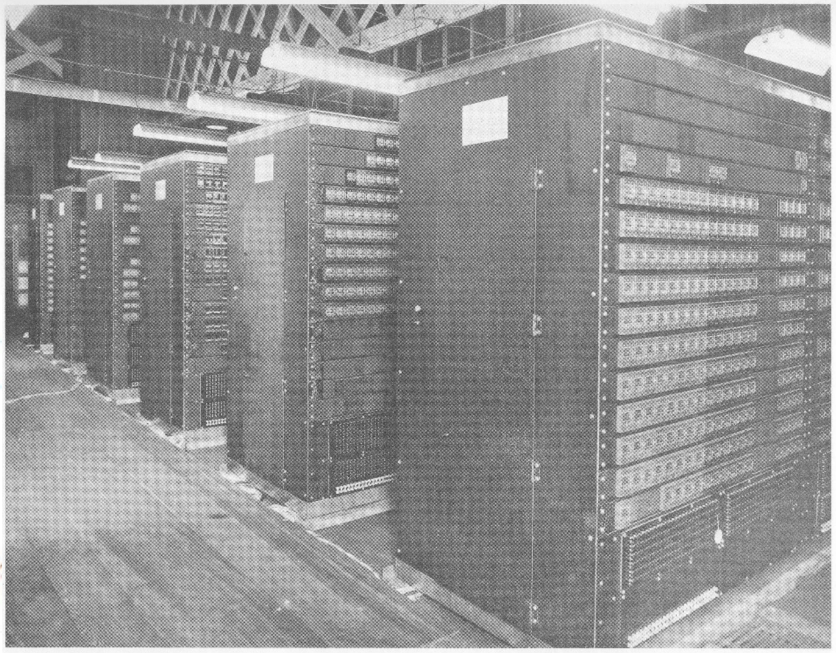


Figure 5 Mark II: Relay Cubicles

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

Classwork 3:

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 3:

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