Introduction to C Programming

Lecture 10: review I

Wenjin Wang

wangwj3@sustech.edu.cn

11-25-2022

Course syllabus

Nr.	Lecture	Date
1	Introduction	2022.9.9
2	Basics	2022.9.16
3	Decision and looping	2022.9.23
4	Array & string	2022.9.30
5	Functions	2022.10.9 (补)
6	Pointer	2022.10.14
7	Self-defined types	2022.10.21
8	I/O	2022.10.28

Nr.	Lecture	Date
9	Head files	2022.11.4
10	Review of lectures I	2022.11.25
11	Review of lectures II	2022.12.2
12	Review of lectures III	2022.12.9
13	AI in C programming	2022.12.16
14	AI in C programming	2022.12.23
15	AI in C programming	2022.12.30
16	Summary	2023.1.6

Course syllabus

Review of lectures I

Review of lectures II

Review of lectures III

1	Introduction	2022.9.9
2	Basics	2022.9.16
3	Decision and looping	2022.9.23
4	Array & string	2022.9.30
5	Functions	2022.10.9 (补)
6	Pointer	2022.10.14
7	Self-defined types	2022.10.21
8	I/O	2022.10.28
9	Head files	2022.11.4

Objective of this lecture

Review the learned lectures

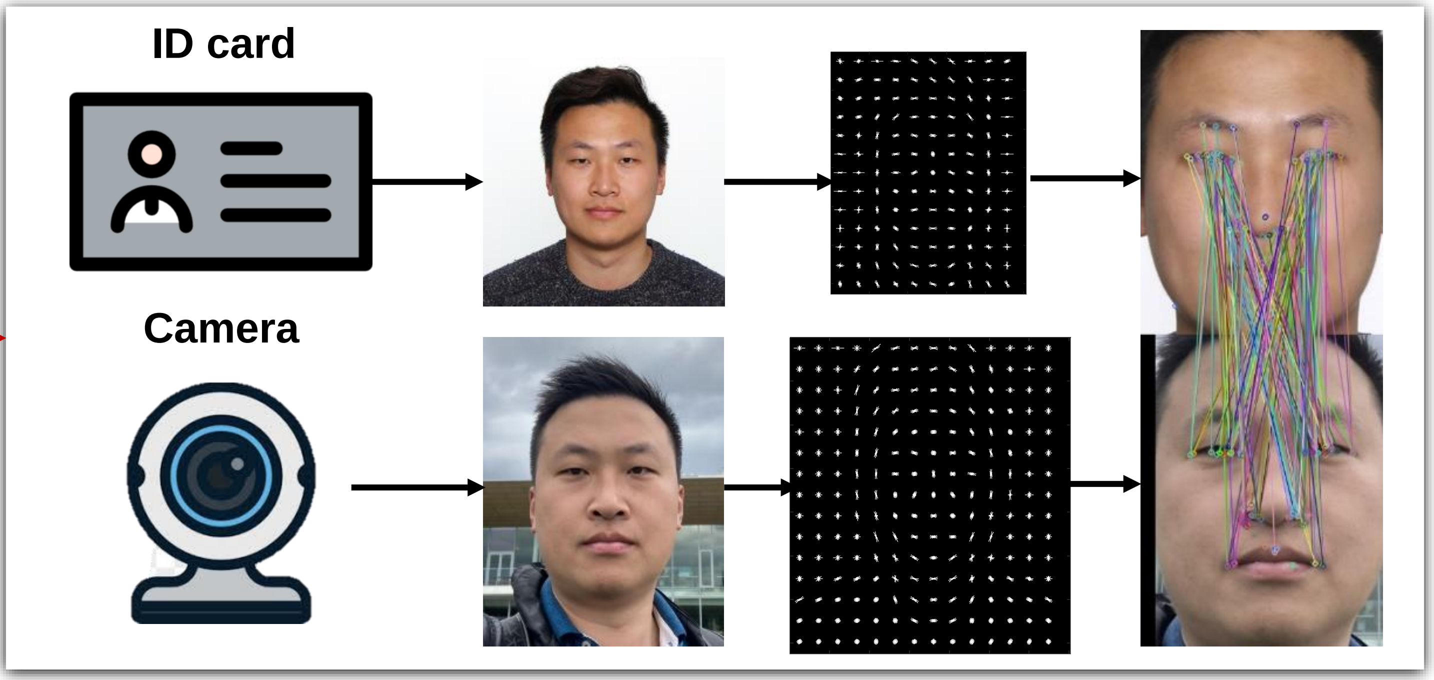
Basics, decision & looping

Content

Content

Machine intelligence is everywhere

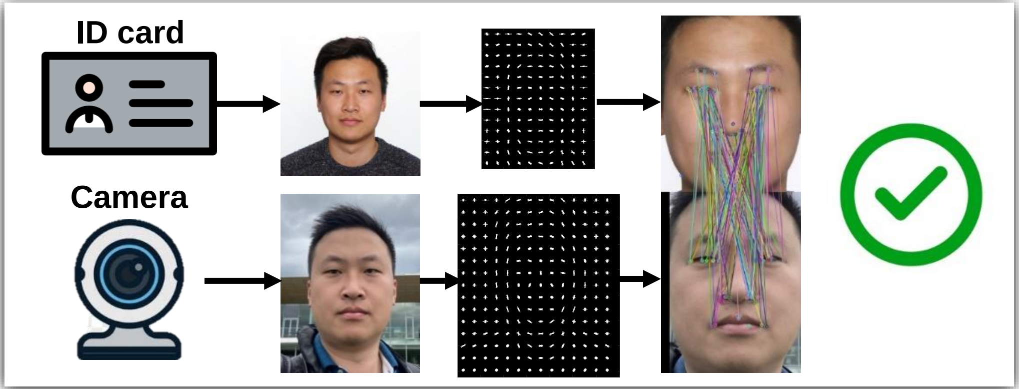
Every time when you enter 深圳北站, following happens…



AI in embedded systems (C)

Machines are controlled by programs

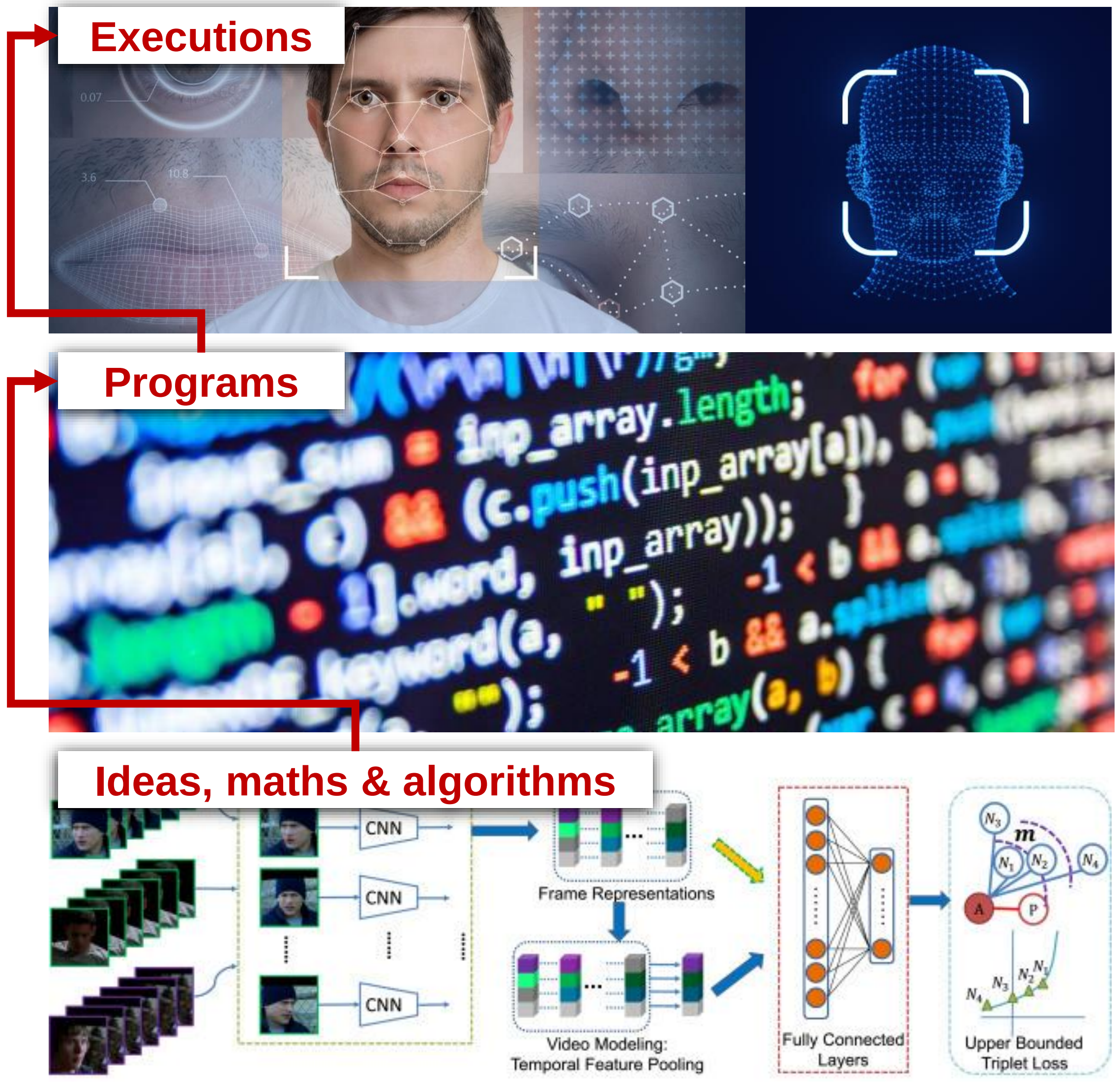
How to build this application?



Sensor + CPU

Database

Gating


Executions
Programs
Ideas, maths & algorithms
CNN
CNN
CNN
Frame Representations
Video Modeling: Temporal Feature Pooling
Fully Connected Layers
Upper Bounded Triplet Loss

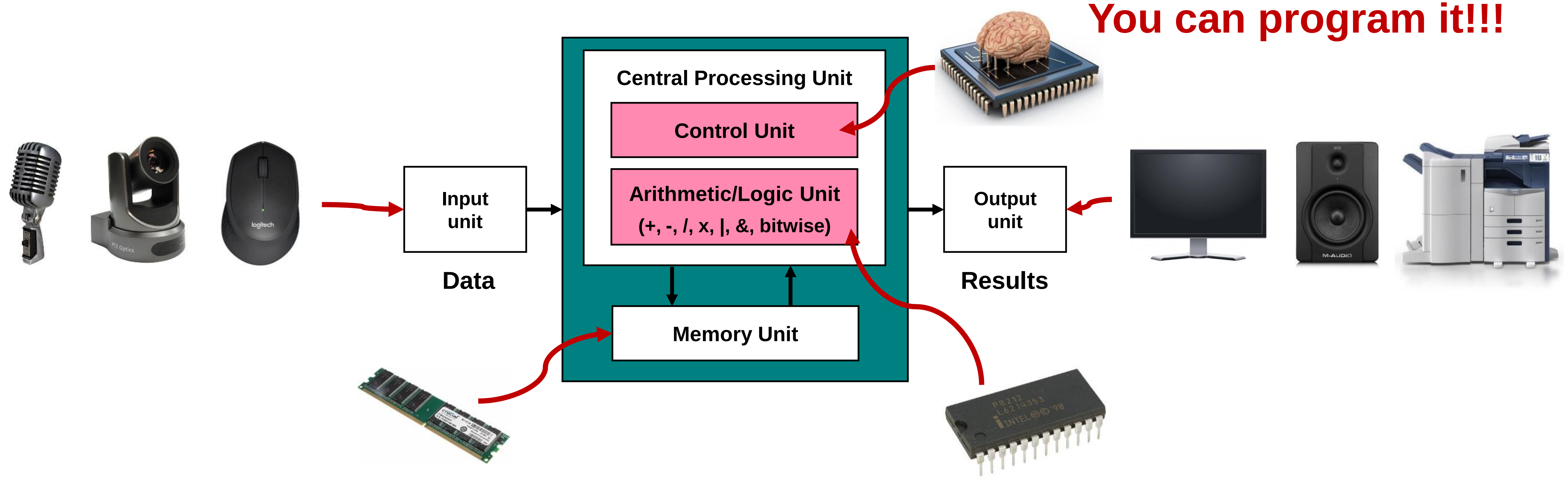
Von Neumann architecture

Machine is programmable

Von Neumann architecture (1946)

冯.诺依曼架构

You can program it!!!



How machine interprets the world?

English is a human language

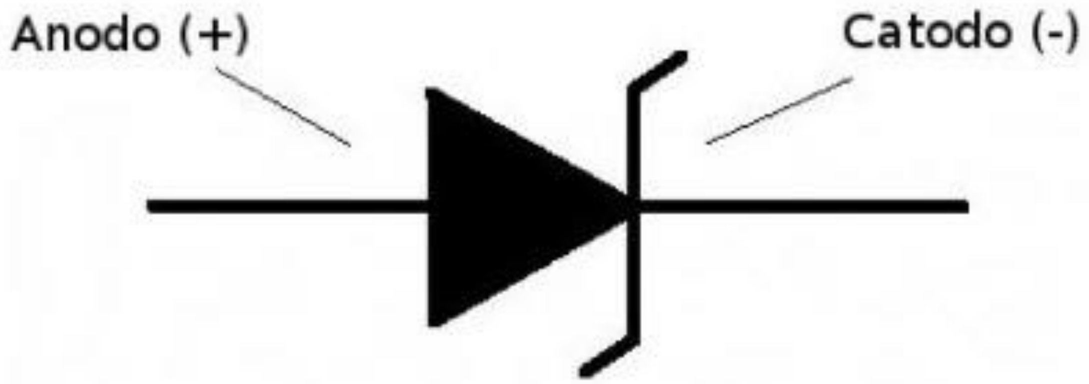
(it has 26 letters: A - Z)

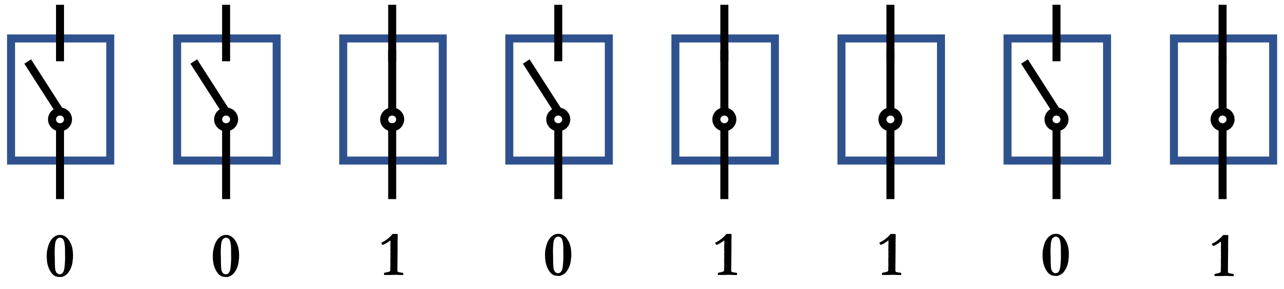
Machine speaks binary language

(it has two states: 0 and 1)

Programming language is a “language”

- A computer is nothing but a vast collection of **electronic switches** (diode) to store information





Programming language is a “language”

Natural language

If you master English, you can talk to American.

Programming language

If you master programming language, you can talk to machines.

C language

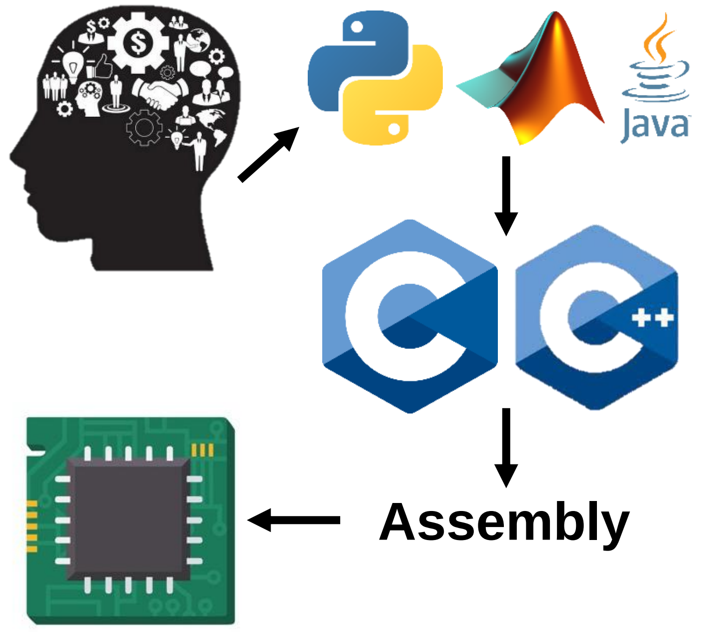
You can talk to machines (0&1) by programming

C Programming

Characteristics of C language

High-level language to human

- High-level syntax, comfortable to be used, e.g. printf(“Hello!”);
- Easy to be structured and extended.



Efficient language to machine

- Way of accessing machine memory is efficient, e.g. bitwise operations, pointers.
- Grammar is close to machine interpretations.

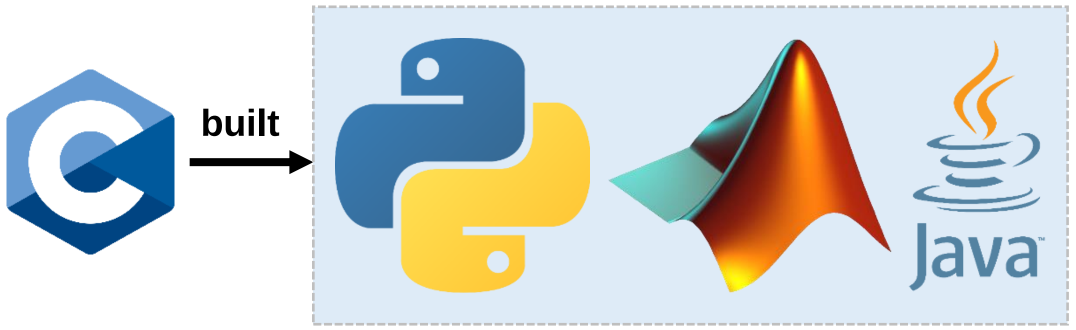
C is desired for edge devices

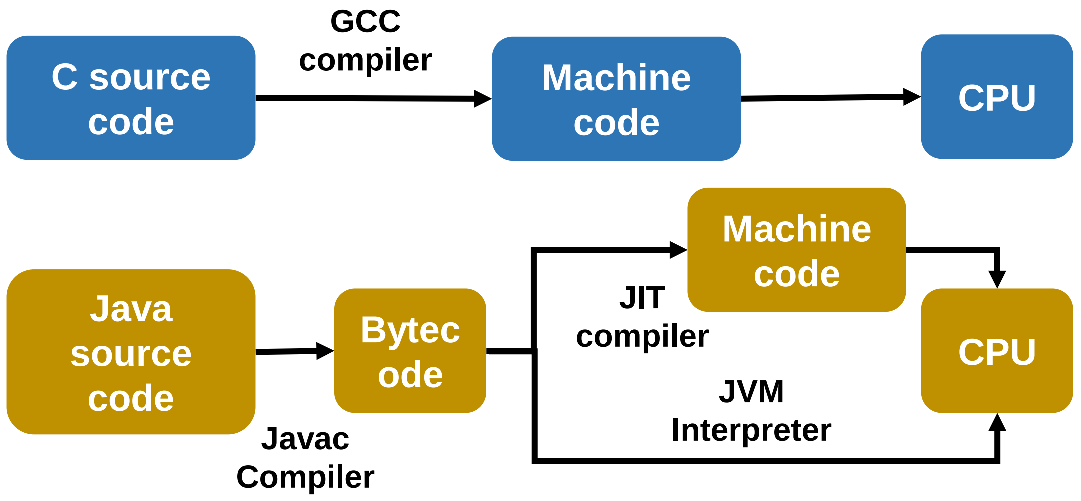
Ubiquitous and scalable for applications

- Can run on Windows/Linux/IOS and various systems.
- Can build programs, compilers and operating systems.

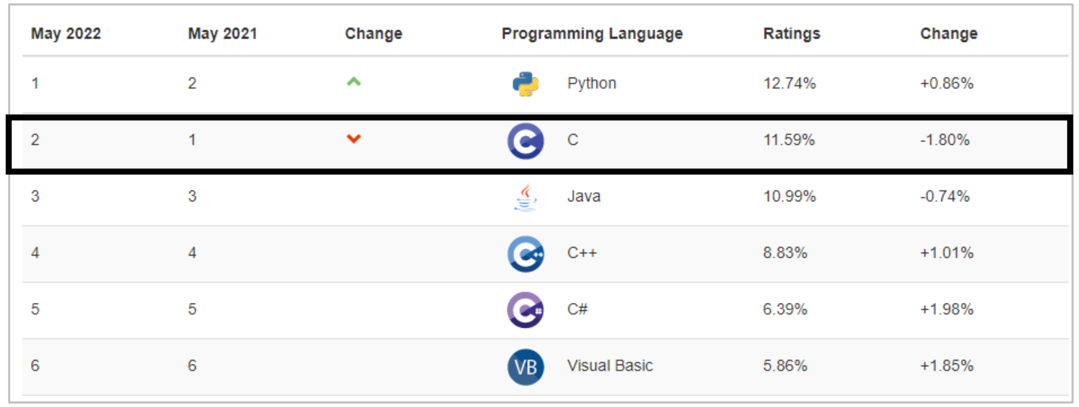
Characteristics of C language

C is the basis of various programming languages





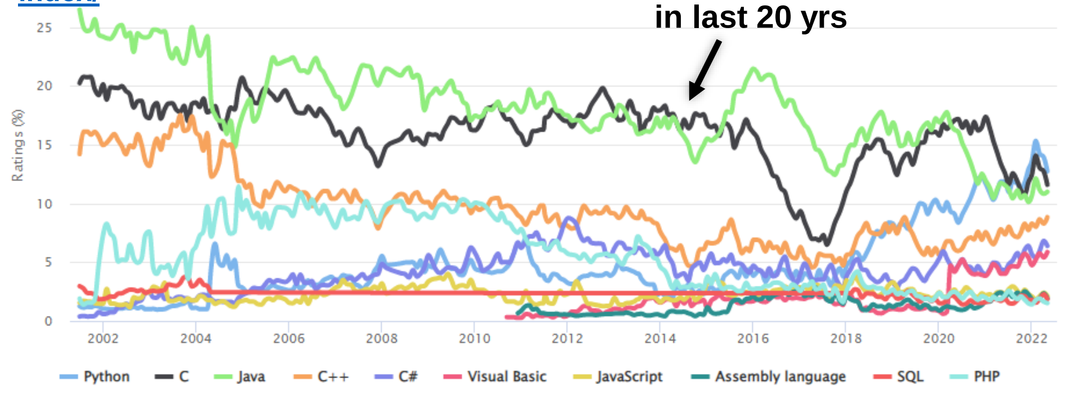
C is a long-live popular language in the industry



May 2022	May 2021	Change	Programming Language	Ratings	Change
1	2	^	Python	12.74%	+0.86%
2	1	˅	C	11.59%	-1.80%
3	3		Java	10.99%	-0.74%
4	4		C++	8.83%	+1.01%
5	5		C#	6.39%	+1.98%
6	6		Visual Basic	5.86%	+1.85%

Source: https://www.tiobe.com/tiobe-index/

Popularity of C is stable in last 20 yrs

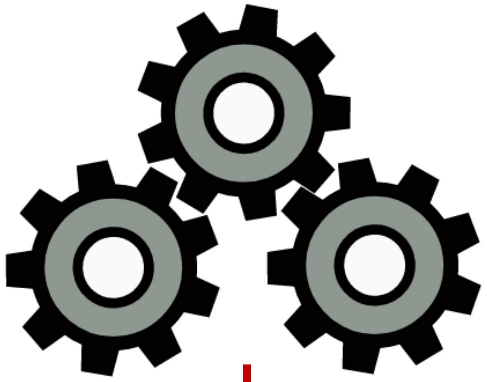


How a C program is executed?

Source code
(human readable program)

```
#include <stdio.h>

int main()
{
    printf("Hello World");

    return 0;
}
```

Compilation
(translation)



Machine code
(executable program)

```
011101101
110010011
001010010
011010100
```

Click to run
(.exe)

HelloWorld.exe

词法解析

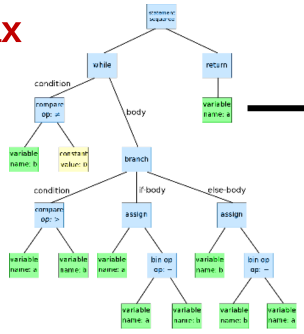
Source code → **Lexical Analysis (scanner)**

token →

← **Request**

句法解析

Syntax Analysis (parser) → **Syntax tree**



→ **Code Generation & optimization** → **Machine code**

<u>**printf("Hello World");**</u>

printf() → pre-defined function
"Hello World" → user-input context

How a C program is executed?

```
#include <stdio.h>

int main()
{
    printf("Hello World\n");
    return 0;
}
```

```
...  // include file
extern void funlockfile (FILE *__stream) __attribute__
((__nothrow__ , __leaf__));
# 868 "/usr/include/stdio.h" 3 4

# 2 "test.c" 2
# 5 "test.c"
int main()
{
    printf("Hello World\n");
    return 0;
}
```

```
main:
.LFB0:
.cfi_startproc
pushq           %rbp
.cfi_def_cfa_offset 16
.cfi_offset 6, -16
movq            %rsp, %rbp
.cfi_def_cfa_register 6
subq            $16, %rsp
```

```
.cfi_def_cfa 7, 8
.cfi_endproc
.LFE0:
.size           main, .-
main
.ident"GCC: (Ubuntu 7.5.0-
3ubuntu1~18.04) 7.5.0"
.section        .note.GNU-
stack,"",@progbits
```

```
3243020 00 00 00 00 00 00 00 00 1b 00 00 00 07 00 00 00
3243040 02 00 00 00 00 00 00 00 70 02 40 00 00 00 00 00
3243060 70 02 00 00 00 00 00 00 20 00 00 00 00 00 00 00
3243100 00 00 00 00 00 00 00 00 08 00 00 00 00 00 00 00
3243120 00 00 00 00 00 00 00 00 2e 00 00 00 07 00 00 00
3243140 02 00 00 00 00 00 00 00 90 02 40 00 00 00 00 00
3243160 90 02 00 00 00 00 00 00 24 00 00 00 00 00 00 00
3243200 00 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00
3243220 00 00 00 00 00 00 00 00 41 00 00 00 07 00 00 00
3243240 02 00 00 00 00 00 00 00 b4 02 40 00 00 00 00 00
3243260 b4 02 00 00 00 00 00 00 20 00 00 00 00 00 00 00
3243300 00 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00
3243320 00 00 00 00 00 00 00 00 4f 00 00 00 04 00 00 00
3243340 42 00 00 00 00 00 00 00 d8 02 40 00 00 00 00 00
```

HelloWorld.exe

Source code (.c .h .cpp)

↓ **Precompiling**
- Expand include
- Add line numbers and file IDs

Source code (.i)

↓ **Compile**
- Lexical analysis
- Semantic analysis
- Code optimization
- generate assembly language (.s)

Assembly code (.s)

↓ **Compilation** → Translate assembly language(.s) into machine language(.o)

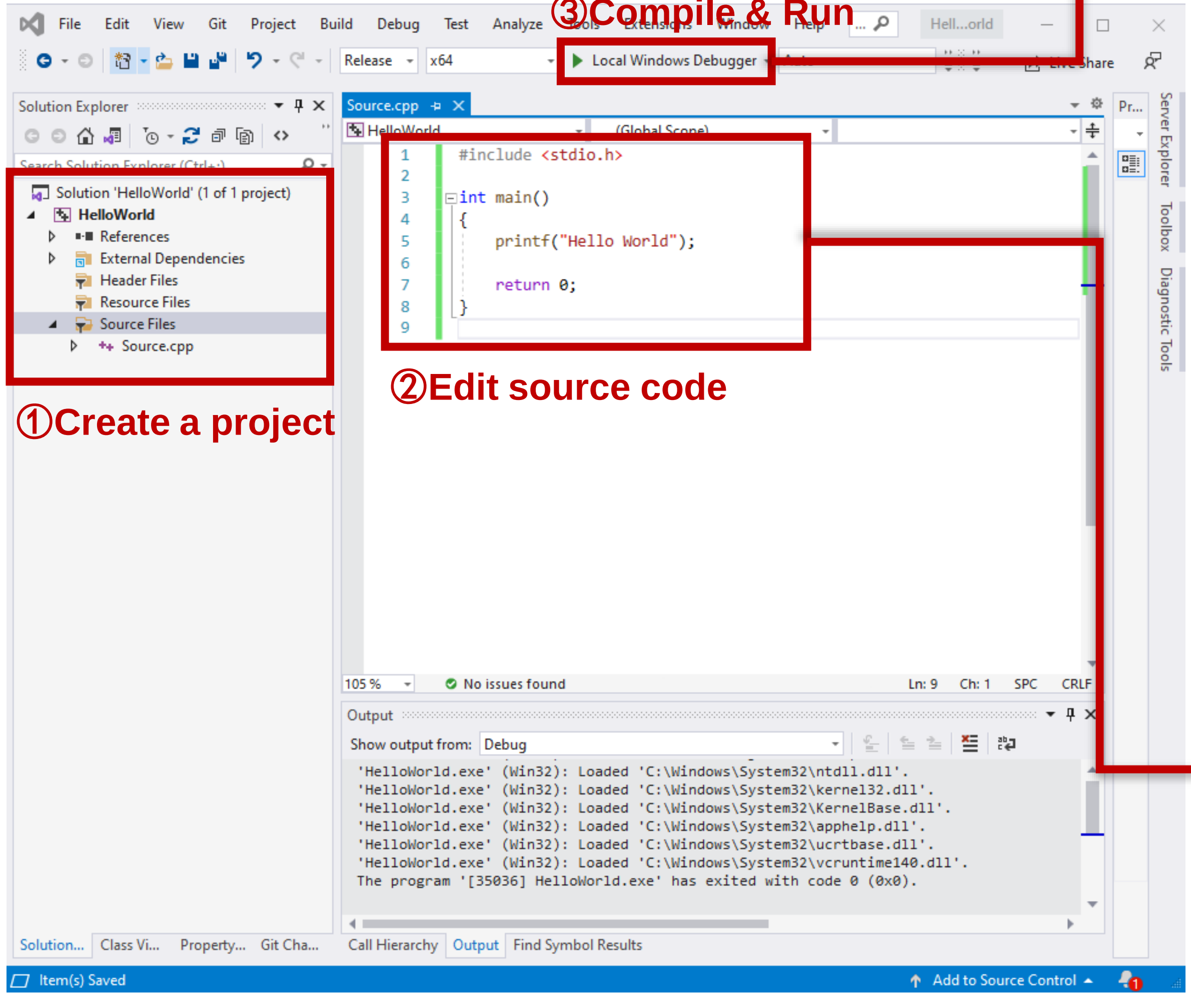
Machine code (.o)

↓ **Link** → Link intermediate files(.o) into executable files(.exe)

Executable file (.exe)

We use Visual Studio IDE (集成开发环境)

Visual Studio IDE


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Hell...orld
Release x64 Local Windows Debugger
Live Share
Solution Explorer
Solution 'HelloWorld' (1 of 1 project)
HelloWorld
References
External Dependencies
Header Files
Resource Files
Source Files
Source.cpp
Source.cpp
HelloWorld (Global Scope)
105 % No issues found Ln: 9 Ch: 1 SPC CRLF
Output
Show output from: Debug
Solution... Class Vi... Property... Git Cha... Call Hierarchy Output Find Symbol Results
Item(s) Saved Add to Source Control
Server Explorer Toolbox Diagnostic Tools

```
#include <stdio.h>

int main()
{
    printf("Hello World");

    return 0;
}
```

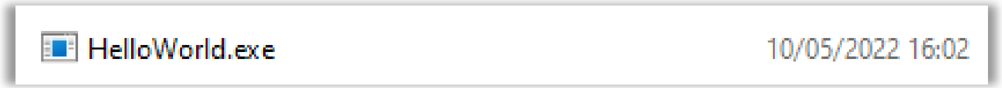
```
'HelloWorld.exe' (Win32): Loaded 'C:\Windows\System32\ntdll.dll'.
'HelloWorld.exe' (Win32): Loaded 'C:\Windows\System32\kernel32.dll'.
'HelloWorld.exe' (Win32): Loaded 'C:\Windows\System32\KernelBase.dll'.
'HelloWorld.exe' (Win32): Loaded 'C:\Windows\System32\apphelp.dll'.
'HelloWorld.exe' (Win32): Loaded 'C:\Windows\System32\ucrtbase.dll'.
'HelloWorld.exe' (Win32): Loaded 'C:\Windows\System32\vcruntime140.dll'.
The program '[35036] HelloWorld.exe' has exited with code 0 (0x0).
```

①Create a project

②Edit source code

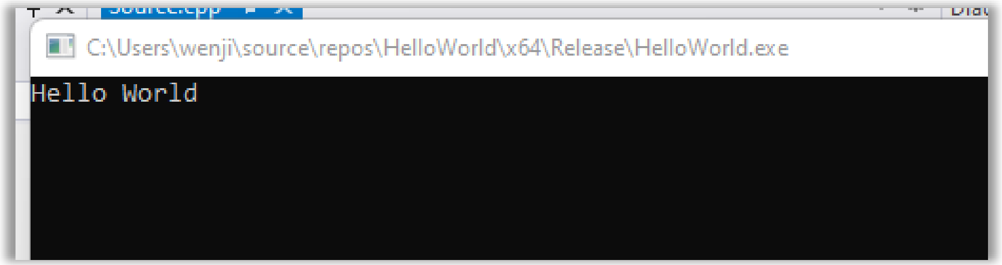
③Compile & Run

Generate executable file



HelloWorld.exe 10/05/2022 16:02

Run the executable file



C:\Users\wenji\source\repos\HelloWorld\x64\Release\HelloWorld.exe

```
Hello World
```

```
#include <stdio.h>

int main()
{
    printf("Hello World");

    return 0;
}
```

Our first C program: Hello world

Exit

Entrance

Do something!

```
#include<stdio.h>
int main()
{
    printf(“Hello World %d!”);
    return 0;
}
```

Our first C program: Hello world

1

```
main()
{
    //do nothing!
}
```

2

```
main()
{
    printf("Hello World!");
    // error, cannot recognize
}
```

3

```
#include<stdio.h>

main()
{
    printf("Hello World!");
}
```

4

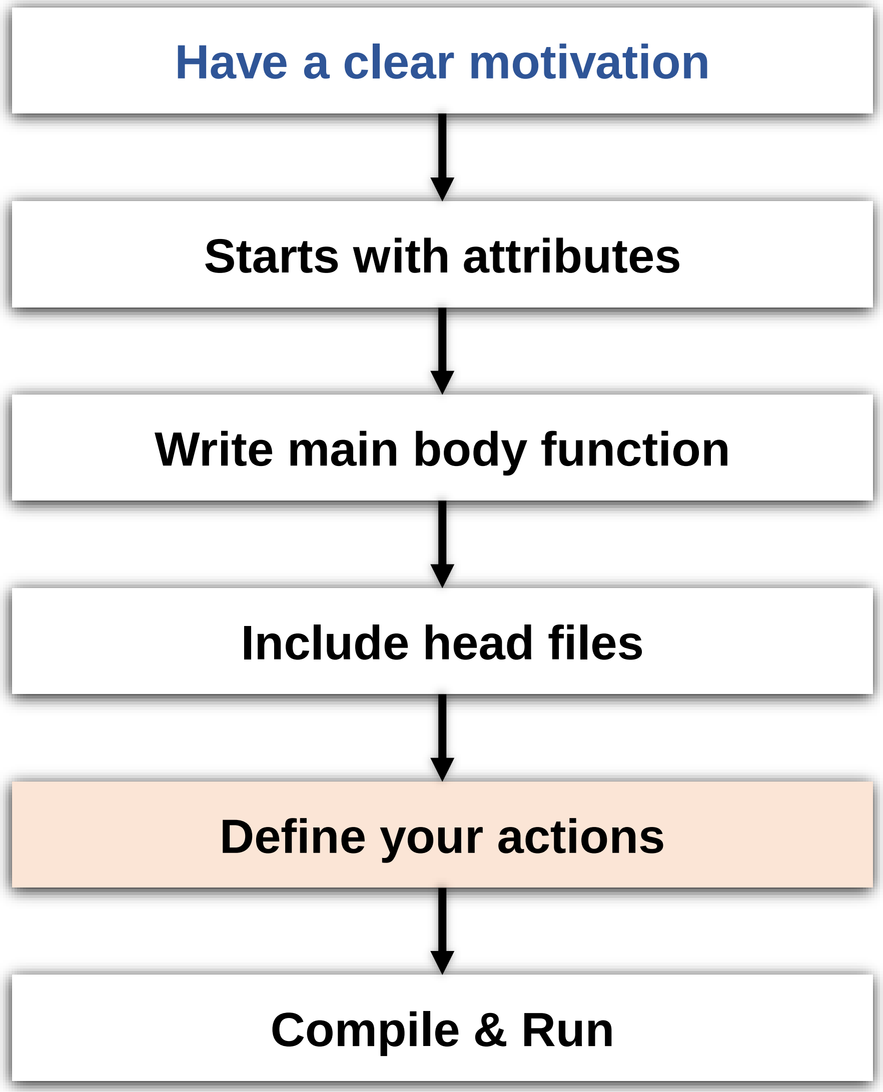
```
#include<stdio.h>

int main()
{
    printf("Hello World!");
    return 0;
}
```

5

```
#include<stdio.h>

int main(int a)
{
    printf("Hello World %d!", a);
    return 0;
}
```

Our first C program: Hello world

```c
/****************************************
* Name:     HelloWorld.c
* Author:   Wenjin Wang
* Date:     5-10-2022
* Abstract: show HelloWorld printing example
* Version:  1.0
* Copyright:SUSTech
****************************************/

#include <stdio.h>

int main()
{
    printf("Hello World"); // This is to
print "Hello World“

    return 0;
}
```



Summary of introduction

- Machine and machine intelligence are everywhere. The way to control machine is by programming.
- C + AI (or domain knowledge) makes you different.
- Programming language allows communication between human and machine. A good language should be friendly to users while still efficient to machines, like C.
- C is a high-level language that is popular and ubiquitous in industry, especially for edge devices.
- Good ways to learn C is practicing with projects, understand the essence instead of memorizing it.

5 Questions

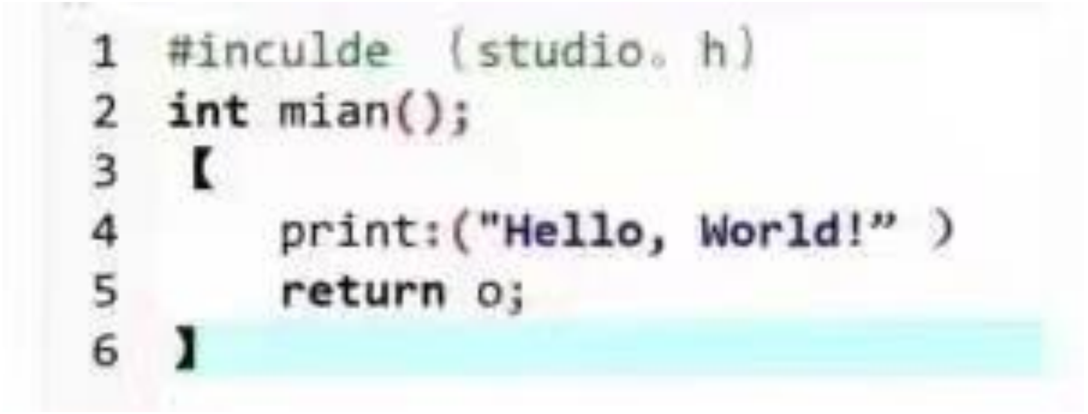
1. C program have an entrance function, what is the name of this function?

2. C program can have two **main** functions. Yes/No

3. You must include a head file to use the **printf** and **scanf**, what is the name of this head file?

4. How a C program is executed? ( )
  A. start from the main function and end when main is returned
  B. start from the first function and end at the last function
  C. start from the first function and end at the first function
  D. start from the main function and end at the last function

5. What are the mistakes in the right C code? Rewrite it!



```
#inculde (studio. h)
int mian();
 【
     print:("Hello, World!” )
     return o;
 】
```

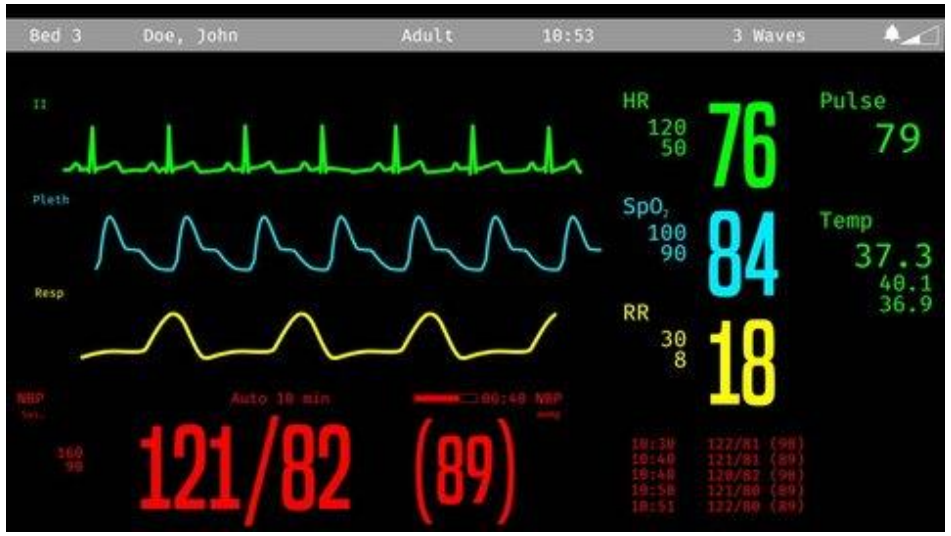
Content

Basics

$$\begin{tikzcd}
\text{User I/O} \arrow[d] \\
\text{Variables} \arrow[d] \\
\text{Operations} \arrow[d] \\
\text{Results}
\end{tikzcd}$$

Data types and operators in life

Data types and operators in life



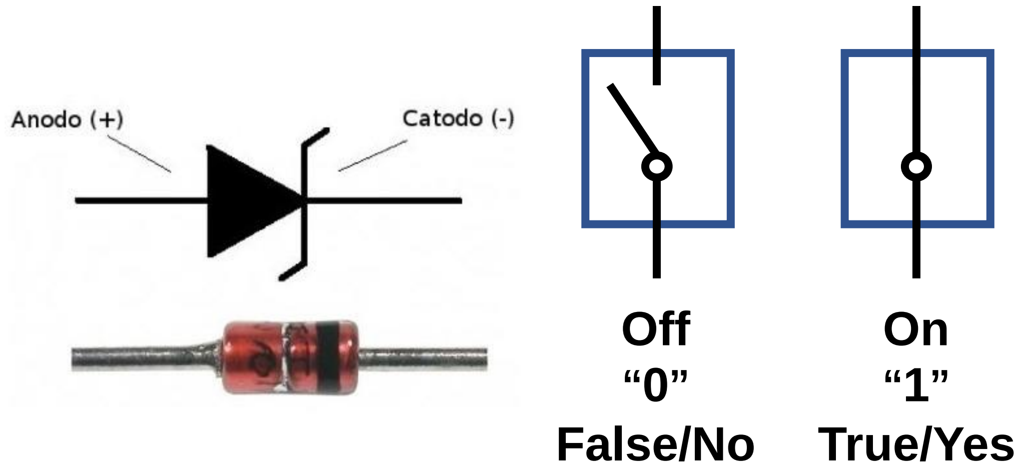
```
string name: Helen
char gender: F
int gestational age: 32 months
int height: 20 cm
int weight: 2 kg
int HR: 160 bpm
int RR: 60 bpm
int SpO2: 96%
```

Bit and byte

Bit (位)

The smallest unit for storage (atomic),

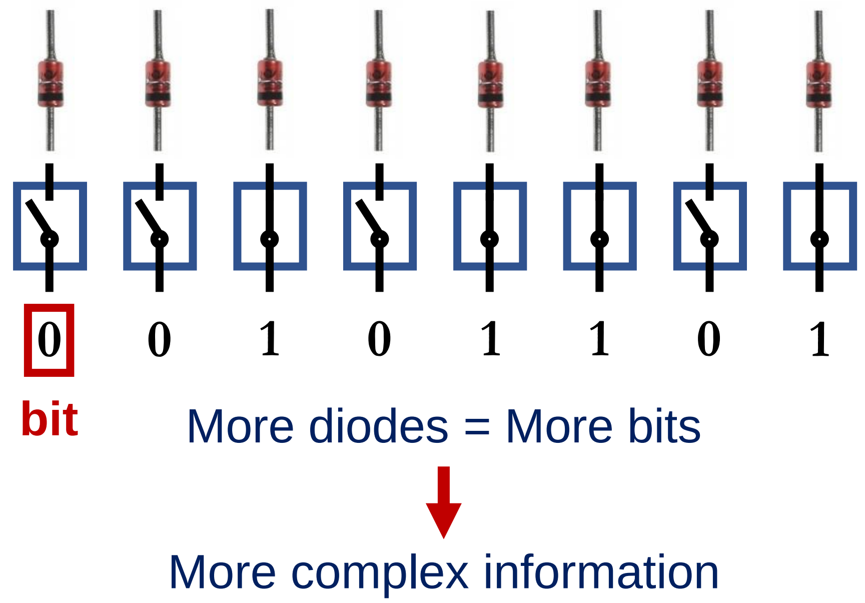
0 or 1



Byte (字节)

The smallest unit for information storage,

1 byte = 8 bits



Decimal numbering system

3 8 4 6

$$= (3 * 10^3) + (8 * 10^2) + (4 * 10^1) + (6 * 10^0)$$

$\downarrow$

Use 10 as basis

Binary numbering system

1 0 0 1 0

$$= (1 * 2^4) + (0 * 2^3) + (0 * 2^2) + (1 * 2^1) + (0 * 2^0)$$

$$= 18$$

$\downarrow$

Use 2 as basis

Decimal to binary

```
2 | 3846   ...... 0
2 | 1923   ...... 1
2 | 961    ...... 1
2 | 480    ...... 0
2 | 240    ...... 0
2 | 120    ...... 1
2 | 60     ...... 1
2 | 30     ...... 1
2 | 15     ...... 1
2 | 7      ...... 1
2 | 3      ...... 1
2 | 1      ...... 1
    0
```

除2取余，直到除数为0

3846

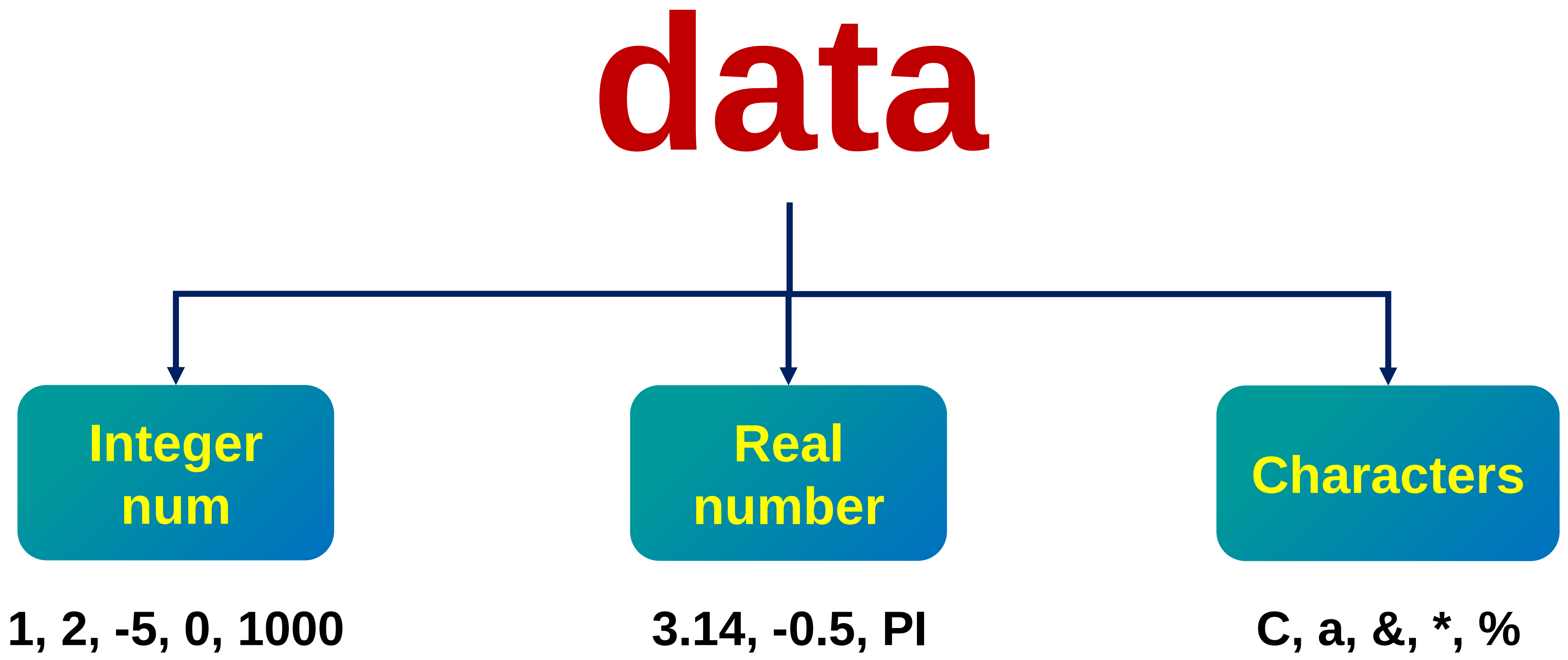
1 1 1 1 0 0 0 0 0 1 1 0

(float) Decimal to binary

$$\begin{tikzcd}
88.8125 \arrow[r] \arrow[d] & 0.8125 \arrow[d] & \\
88 \arrow[d] & 1101 \arrow[dl] & \\
1011000 \arrow[r] & 1011000.1101 &
\end{tikzcd}$$

$$\begin{array}{lll}
0.8125 & \times 2 - 1 & \ldots\ldots 1 \\
= 0.625 & \times 2 - 1 & \ldots\ldots 1 \\
= 0.25 & \times 2 & \ldots\ldots 0 \\
= 0.5 & \times 2 - 1 & \ldots\ldots 1 \\
= 0 & &
\end{array}$$

Use byte to store data



Data types

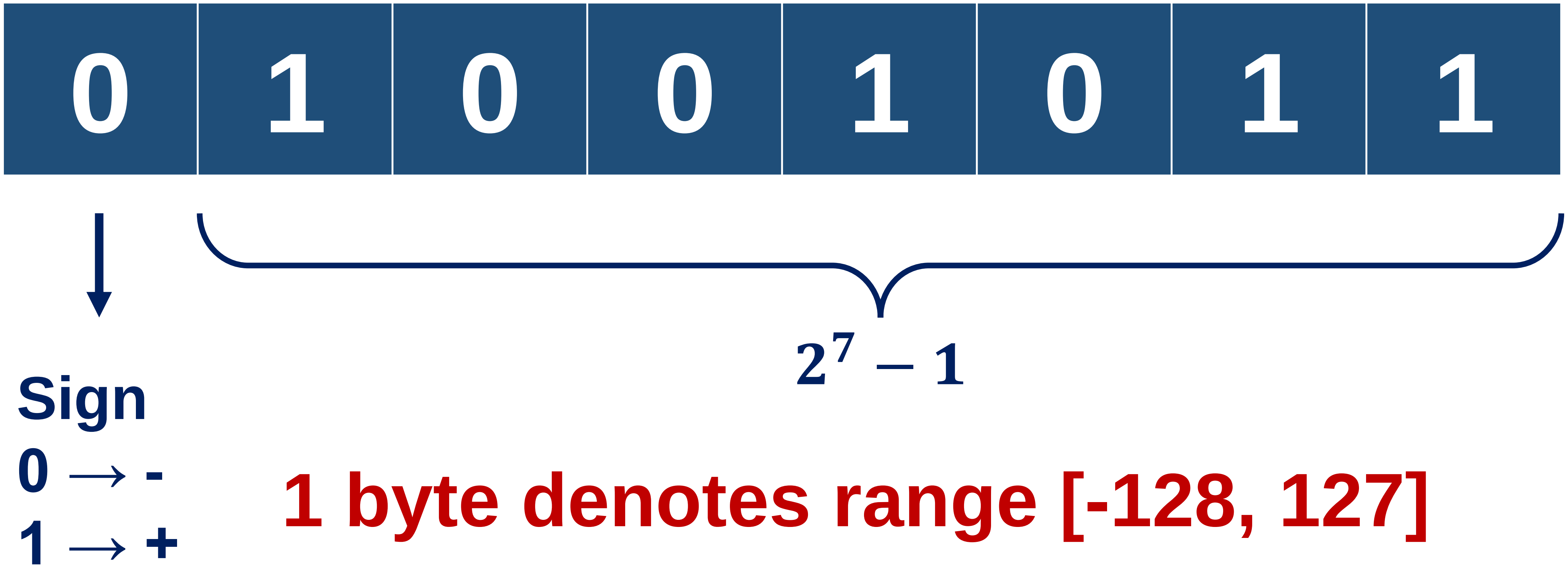
char	int	float/double
Name (‘5’)	Salary (5)	Height
Place	Age	Weight
Food	Student ID	Distance
Word	Heart rate	PI

Data types

	Storage size	Number of bits	Value range	Example
char	1 byte	8 bits	0-255 (128 characters)	A, B, Z, &, $, %
int	4 byte	4 x 8 = 32 bits	-2E+31 to 2E+31-1	20220901
float	4 byte	4 x 8 = 32 bits	-3.4E+38 to 3.4E+38-1	3.1415926
double	8 byte	8 x 8 = 64 bits	2.3E-308 to 1.7E+308-1	3.14159265359
void	0 byte	0 bit	-	-

- **Signed int**, use 1 bit to denote sign, range: -2E+31 to 2E+31-1
- **Unsigned int**, use all bits to denote value, range: 0 to 2E+32-1

Use byte to store integer number



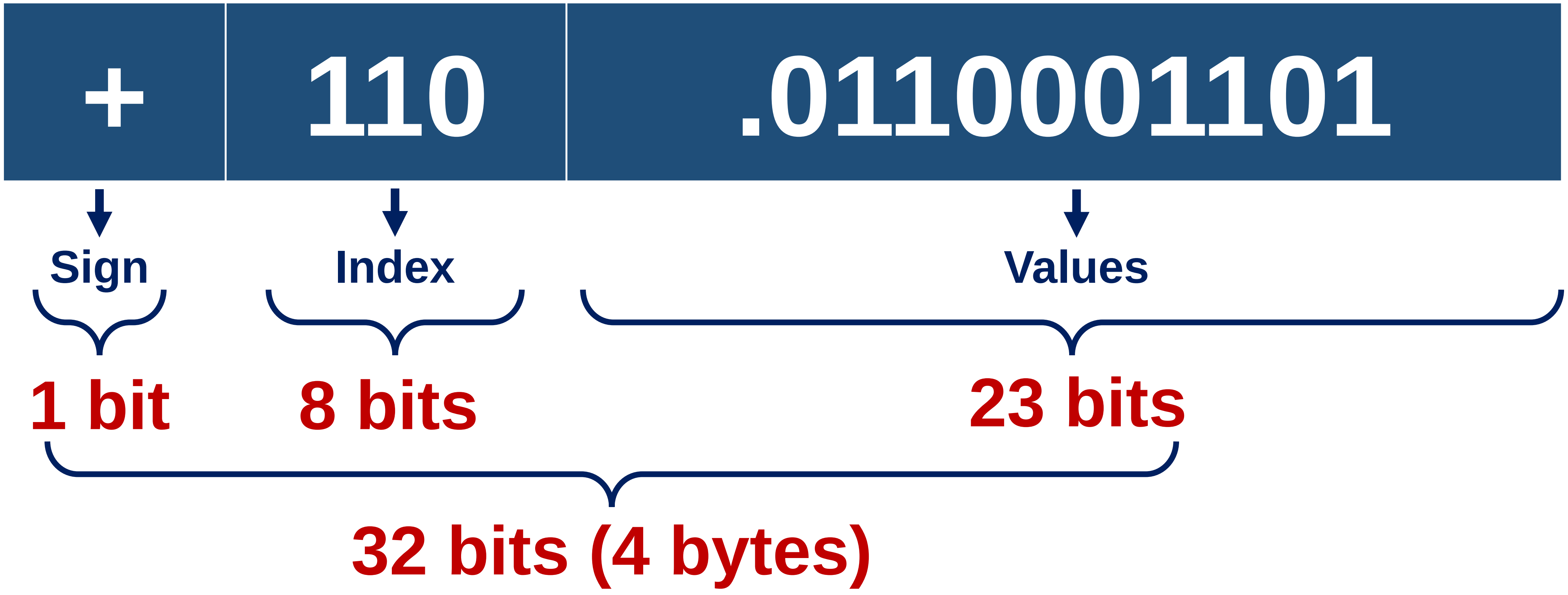
1 byte denotes range [-128, 127]

Use byte to store real number

Real number = rational number (10, -0.23) + irrational number (PI, $\sqrt{2}$)

Use 2 as basis

How to use byte to denote 88.8125? $88.8125 = 1011000.1101 = 1.0110001101 \times 2^6$



Use byte to store character(s)

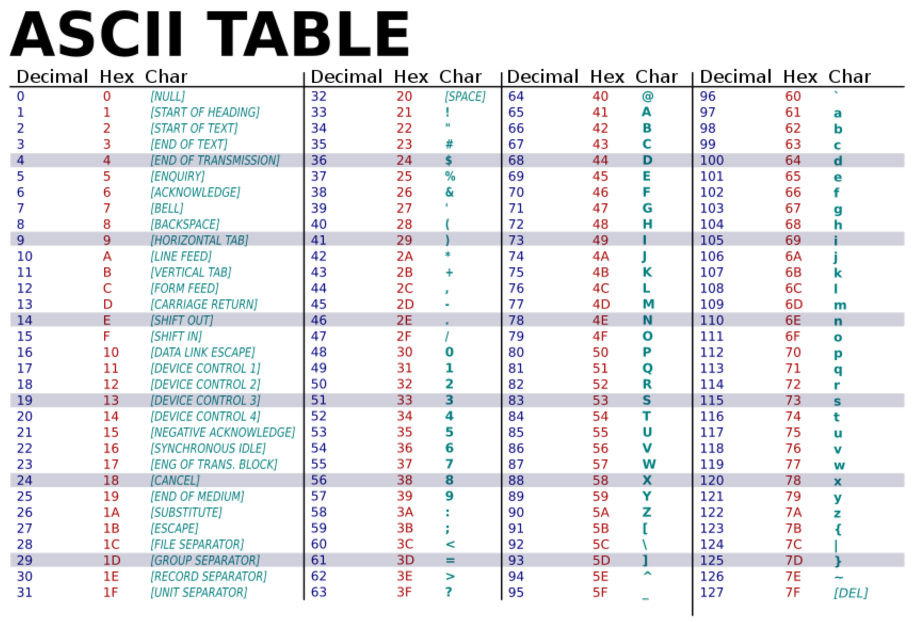
Characters are A, B, C, &, $, %, etc.

0	1	0	0	0	0	0	1

= 65 → ‘A’

find in ASCII table

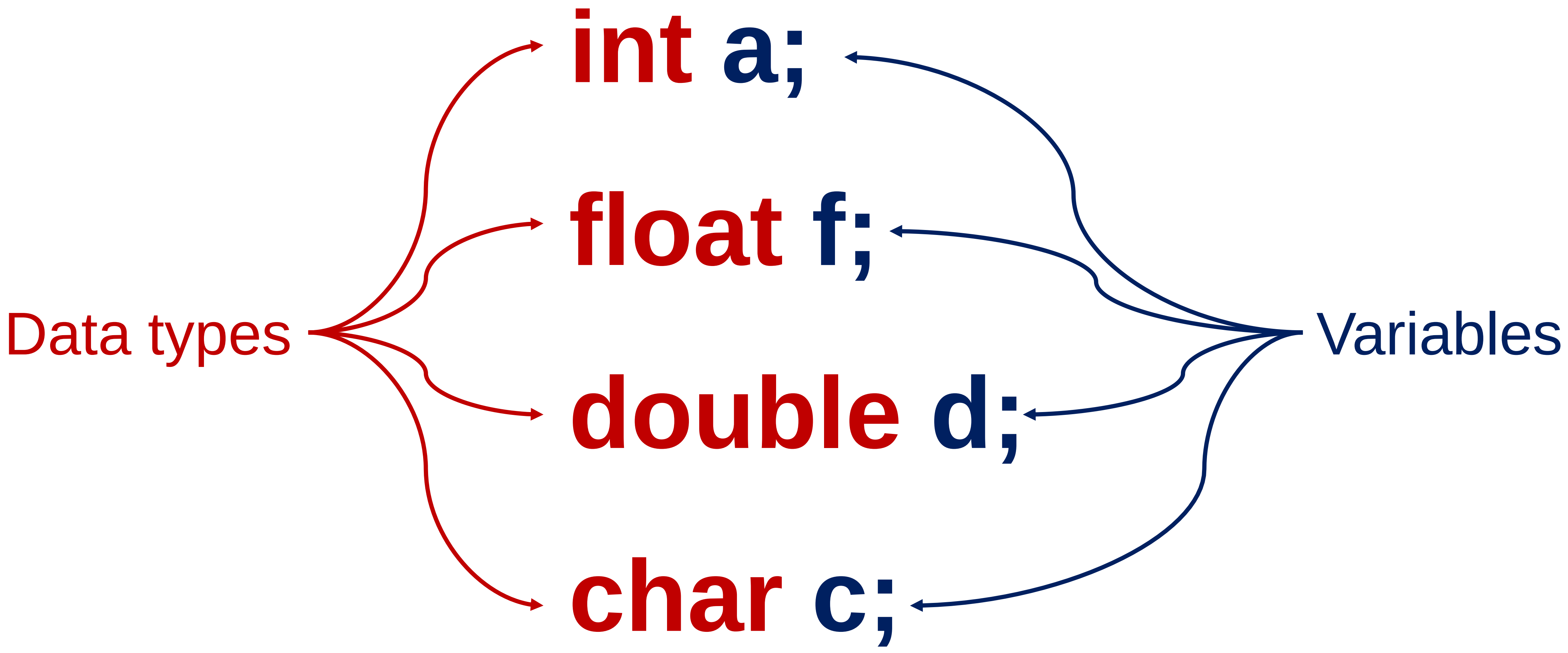
(American National Standard Code for Information Interchange)



ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Data types declare variables



```
int a;
float f;
double d;
char c;
```

Use variables to hold data

Variables are placeholders for values, each variable has a **type** defined. The type determines how it is stored and how much space (bits) it needs in machine.

```
int num; //声明
num = 5; //赋值
printf(“num = %d”, num);
```

```
int num = 5; //声明+赋值
printf(“num = %d”, num);
```

Variables

A variable name can ONLY be defined once, but its value can be set multiple times!

```
int num = 5; //声明+赋值
printf(“num = %d”, num);
```

✓
```
num = 10; //重新赋值
printf(“num = %d”, num);
```

```
int num = 5; //声明+赋值
printf(“num = %d”, num);
```

✗
```
int num = 10; //重定义
printf(“num = %d”, num);
```

Variables

Declare variables with same type jointly

```
int a = 3;
int b = 4;
int c = 100;
```

⬇

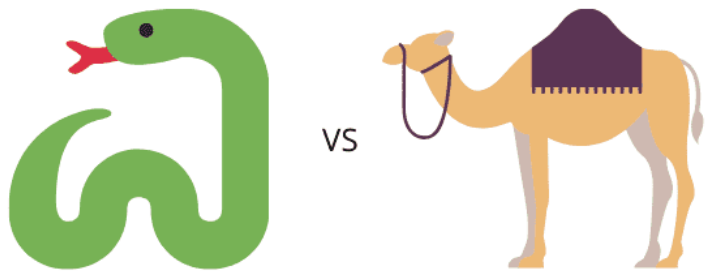
```
int a = 3, b = 4, c = 100;
```

Cast variables（强制转换）

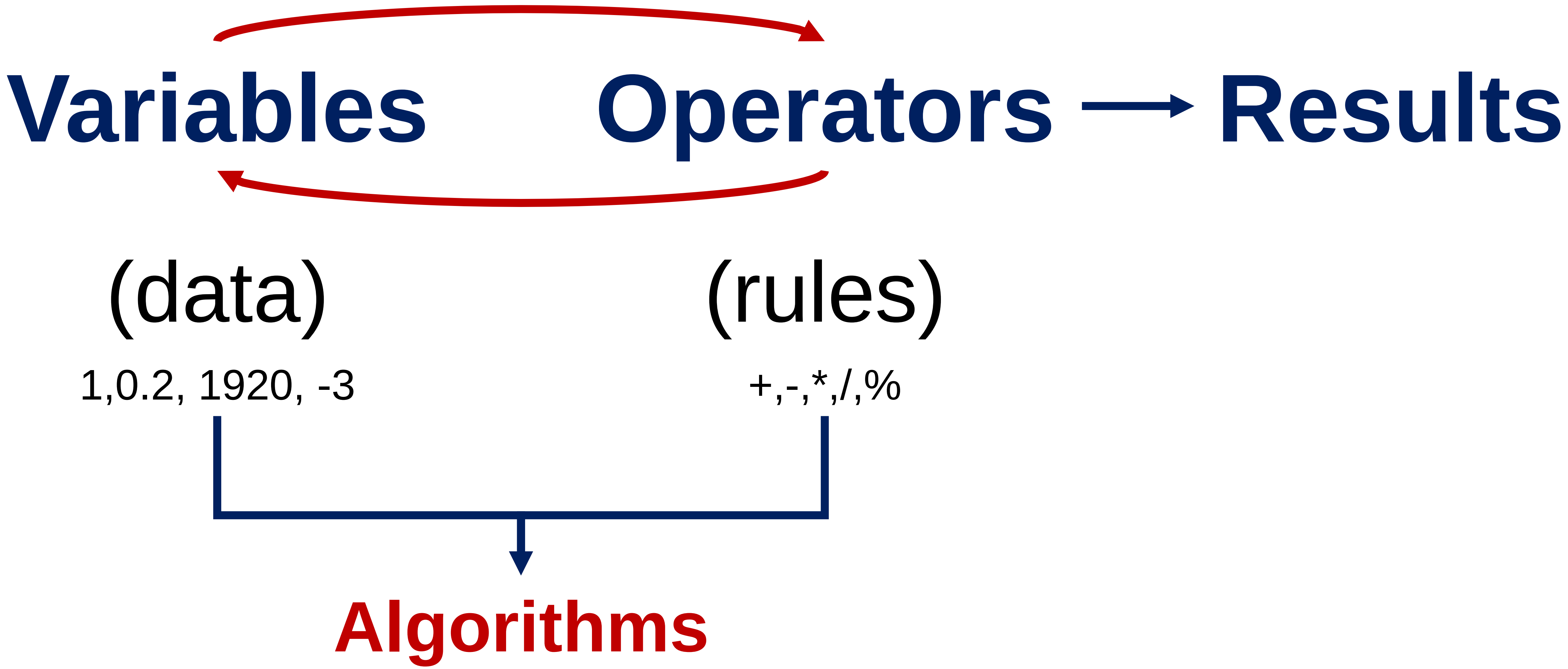
```
float x;
int y = 3;
x = (float) y;
```

Rules to name variables

- **Keywords** are reserved by C, cannot be used!
- Variable names must be **unique**!
- Variable names should be **readable, meaningful and consistent**.
  - UpperCamelCase
  - lowerCamelCase
  - snake_case



Operators


Variables
Operators
Results
(data)
(rules)
1,0.2, 1920, -3
+,-,*,/,%
Algorithms

Operators

Operator is a symbol that tells compiler to perform specific mathematical or logical operations.

Arithmetic Operators（数学）	Relational Operators（关系）	Logical Operators（逻辑）	Assignment Operators（赋值）	Misc Operators（其它）
+,-,*,/,%	>,>=,!=,<	&&, ||, !	=, +=, -=, *=	sizeof(),&,?

Arithmetic operators

Define two variables: int A = 5, B = 3;

Operators	Description	Example
+	Add two variables	A + B = 8
-	Subtract two variables	A – B = 2
*****	Multiply two variables	A * B = 15
/	Divide two variables	A / B = 1
%	Take the reminder **(only for int!)**	A % B = 2
++	Increment by adding 1	A++ = 6
--	Decrement by subtracting 1	A-- = 4

Arithmetic operators

More examples on different data types

Operators	**int A = 10, B = 20;**	**float A = 13, B = 6;**
+	A + B = 30	A + B = 19
-	A – B = -10	A – B = 7
*****	A * B = 200	A * B = 78
/	A / B = 0	A / B = 2.166667
%	A % B = 10	**A % B = ? (wrong!)**
++	A++ = 11	A++ = 14
--	A-- = 9	A-- = 12

Arithmetic operators

Post-increment
(后加)
A++

```
int A = 20;
int B = A++;
printf("A = %d\n", A);
printf("B = %d\n", B);
```

Pre-increment
(先加)
++A

```
int A = 20;
int B = ++A;
printf("A = %d\n", A);
printf("B = %d\n", B);
```

Relational operators

Define two variables: int A = 5, B = 3;

Operators	Description	Example
==	Check if two variables are equal	A==B = 0 (false)
!=	Check if two variables are unequal	A != B = 1 (true)
>	Check if A is larger than B	A > B = 1 (true)
<	Check if A is smaller than B	A < B = 0 (false)
>=	Check if A is larger or equal than B	A >= B = 1 (true)
<=	Check if A is smaller or equal than B	A <= B = 0 (false)

Relational operators

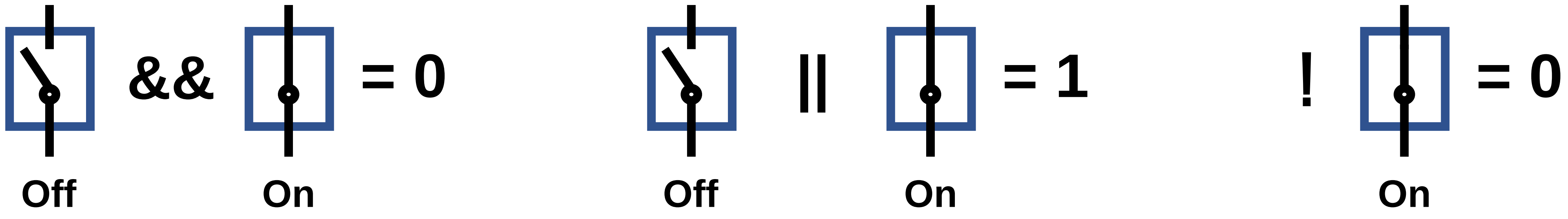
More examples on different data types

Operators	**float A = 3.5, B = 3.5;**	**char A = ‘A’, B = ‘B’;**
==	A==B = 1 (true)	A==B = 0 (false)
!=	A != B = 0 (false)	A != B = 1 (true)
>	A > B = 0 (false)	A > B = 0 (false)
<	A < B = 0 (false)	A < B = 1 (true)
>=	A >= B = 1 (true)	A >= B = 0 (false)
<=	A <= B = 1 (true)	A <= B = 0 (true)

Logical operators

Define two variables: int A = 0, B = 1;

Operators	Description	Example
&&（与）	AND operator, if both are on, then on	A&&B = 0 (false)
||（或）	OR operator, if any is on, then on	A ||B = 1 (true)
!（非）	NOT operator, turn opposite	!A = 1 (true) !B = 0 (false)



Assignment operators

Define two variables: int A = 5, B = 3;

Operators	Description	Example
=	Simple assignment	B = B + A = 8
+=	Add and assign	B += A is B = B + A = 8
-=	Subtract and assign	B -= A is B = B – A = -2
***=**	Multiply and assign	B *= A is B = B * A = 15
/=	Divide and assign	B /= A is B = B / A = 0
%=	Modulus and assign	B %= A is B = B % A = 3

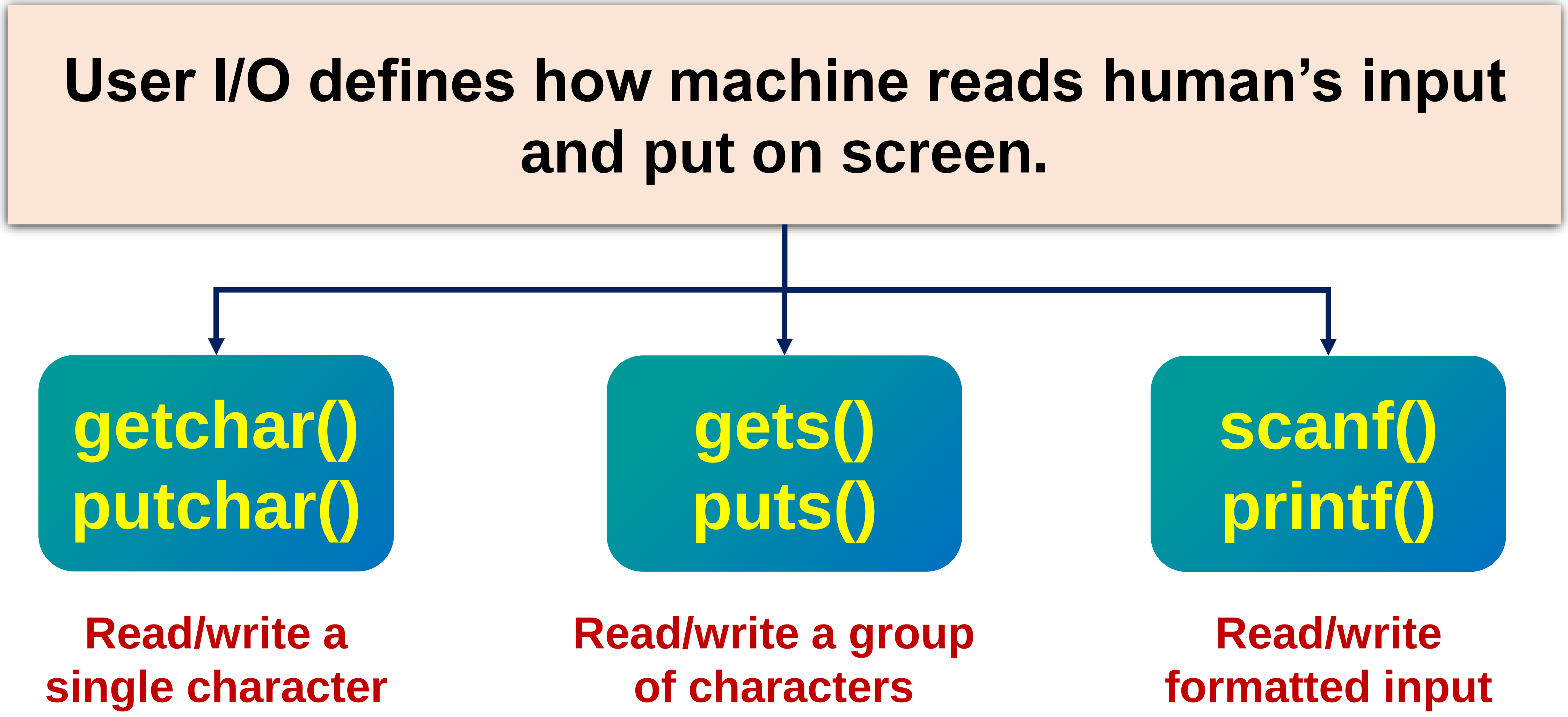
Miscellaneous operators

Define a variable: int A = 10; double B = -1.5;

Operator	Description	Example
sizeof()	Return the size of variable (number of bytes)	sizeof(A) = 4 sizeof(B) = 8
&	Return the address of variable	&A = a84ff7d0 &B = b0affc20
?	Conditional expression	int flag = A>0 ? 1:0;
*****	Pointer points to a variable	*A, *B

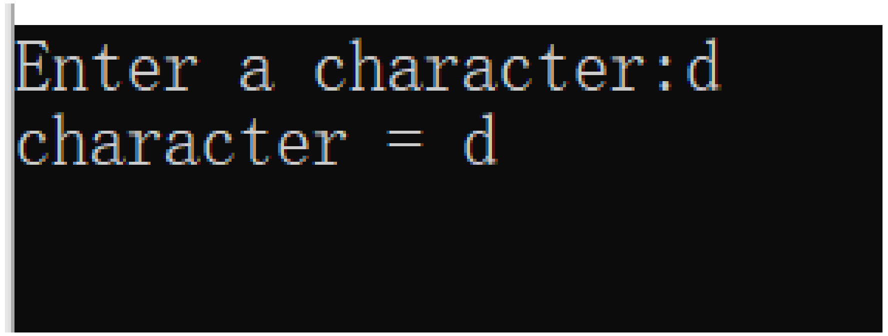
Few other important operators supported by C Language.

User I/O


User I/O defines how machine reads human's input and put on screen.
getchar()
putchar()
gets()
puts()
scanf()
printf()
Read/write a single character
Read/write a group of characters
Read/write formatted input

getchar() and putchar()

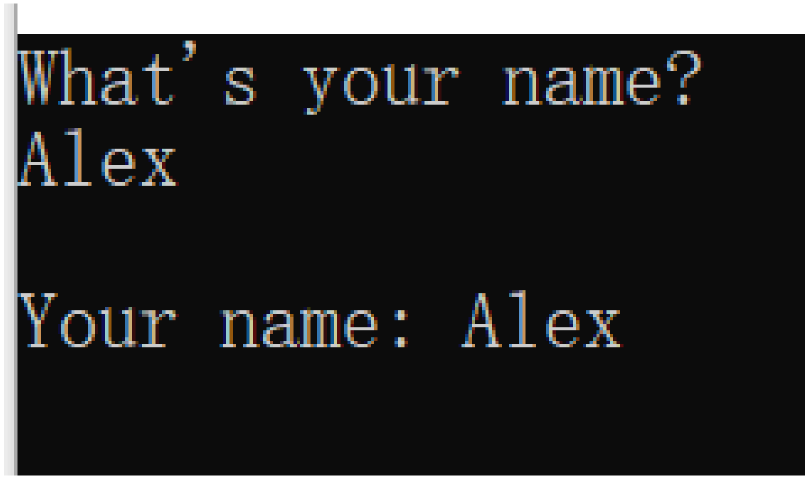
```c
printf("Enter a character:");
int character = getchar();

printf("character = ");
putchar(character);
```



```
Enter a character:d
character = d
```

gets() and puts()

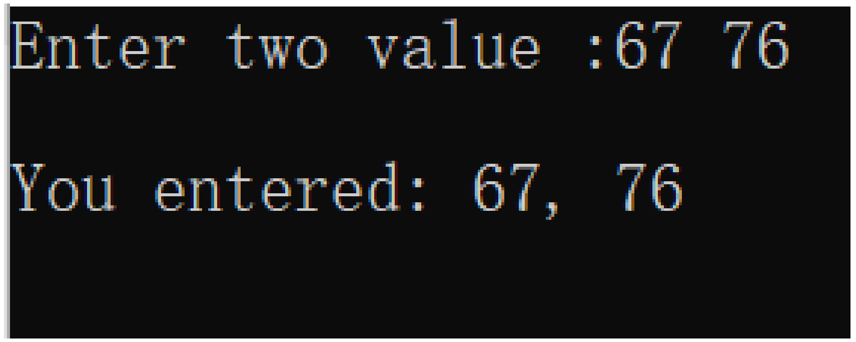
```
char str[20];
printf("What's your name?\n");
gets(str);

printf("\nYour name: ");
puts(str);
```



```
What's your name?
Alex

Your name: Alex
```

scanf() and printf()

```
char str[100];
int i;
printf("Enter two value :");
scanf("%s %d", str, &i);
printf("\nYou entered: %s, %d", str, i);
```



```
Enter two value :67 76

You entered: 67, 76
```

Formatted by specifiers (转义字符)

-	%d	int
-	%f	float
-	%c	char

Step-by-step review

1

```
main()
{
   //do nothing!
}
```

2

```
main()
{
   int a, b;
}
```

3

```
#include<stdio.h>
main()
{
    int a, b;
    printf("Enter two numbers:");
    scanf("%d, %d", &a, &b);
}
```

4

```
#include<stdio.h>
main()
{
    int a, b;
    printf("Enter two numbers:");
    scanf("%d, %d", &a, &b);

    int c = a + b;
}
```

5

```
#include<stdio.h>
main()
{
    int a, b;
    printf("Enter two numbers:");
    scanf("%d, %d", &a, &b);

    int c = a + b;
    printf("a + b = %d\n", c);
}
```

Summary

- Machine uses **bit** (0 and 1) and **byte** (group of bits) to store information
- Different **data types** (int, float, double, char) can be used for declaring and initializing variables based on what you need
- Variables can be used for operations/calculations using **pre-defined operators**
- Five basic operations provided by C: **arithmetic, relational, logical, assignment, Misc**
- Users can interact with the machine using **I/O functions**

5 Questions

1. One byte has how many bits? ()

A. 2　　B. 8　　C. 10　　D. 16

2. What is the output of below printf? ()

```
int main()
	{
	int i = 10;
	int a = i++;
	int b = ++i;
	printf("% d,% d\n", a, b);
	return 0;
}
```

A. 10, 11　B. 10, 12　C. 11, 11　D. 11, 12

5 Questions

3. In below operators, which one requires the operated variables to be integer? ()
A. / B. ++ C. *= D. %

4. Please convert the 1001011 from binary to decimal, and convert 83 from decimal to binary, and writing down the procedure of conversion!

5. In below operators, which one can get the address of a variable? ()
A. & B. && C. ! D. *

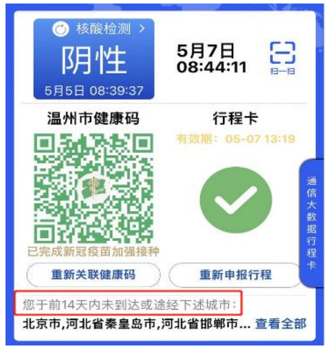
1 0 0 1 0 1 1

$2^6 + 2^3 + 2^1 + 2^0 = 75$

2	83	1	
2	41	1	
2	20	0	
2	10	0	
2	5	1	
2	2	0	
2	1	1	1010011
	0		

Content

Decision-making in life



核酸检测 >
阴性
5月5日 08:39:37

5月7日
08:44:11

扫一扫

温州市健康码

已完成新冠疫苗加强接种

行程卡

有效期：05-07 13:19

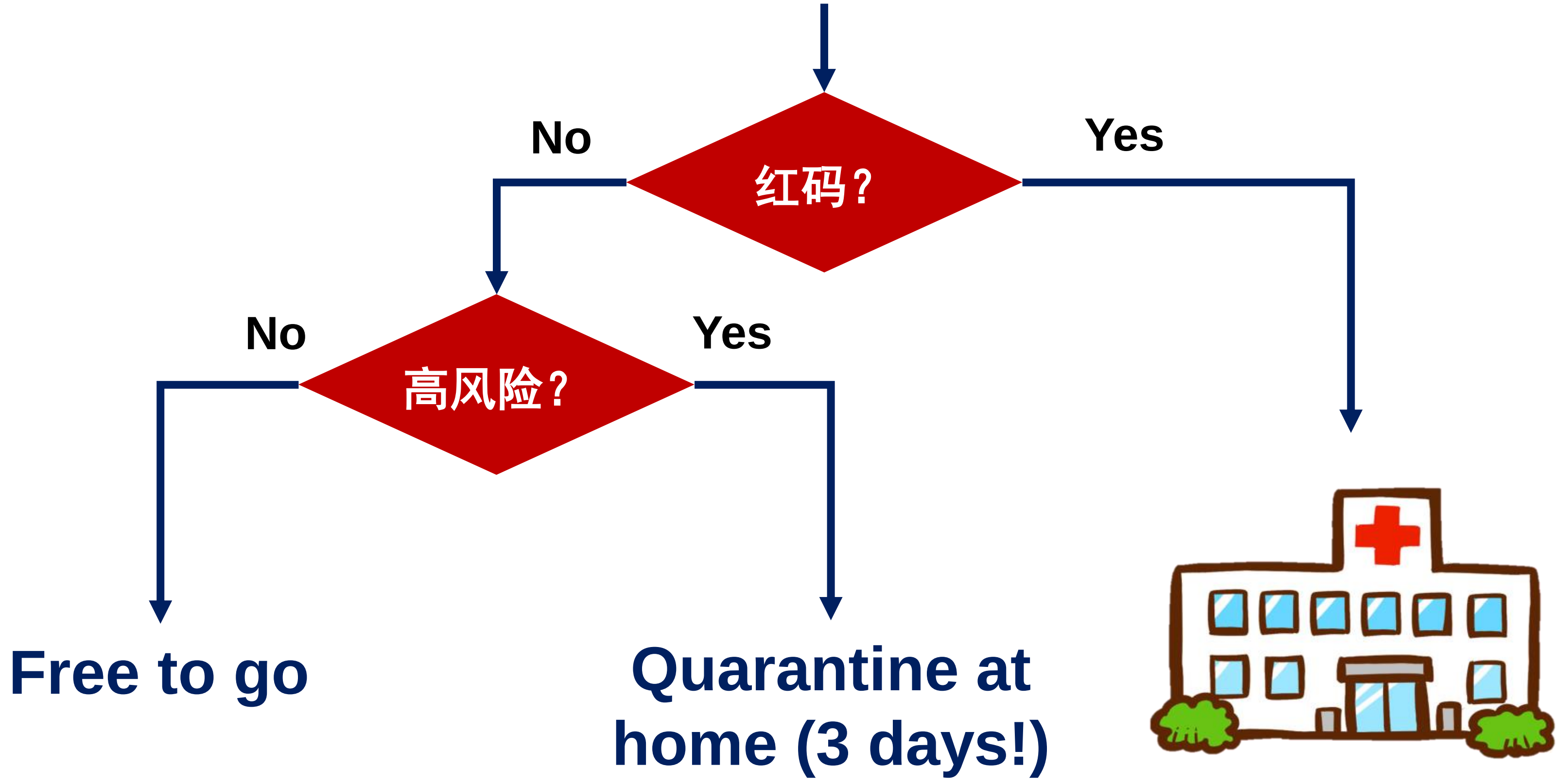
通信大数据行程卡

重新关联健康码

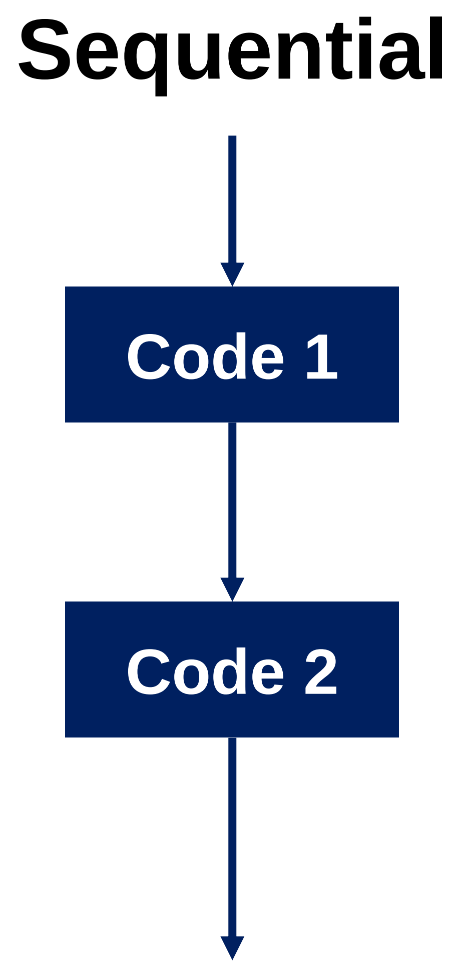
重新申报行程

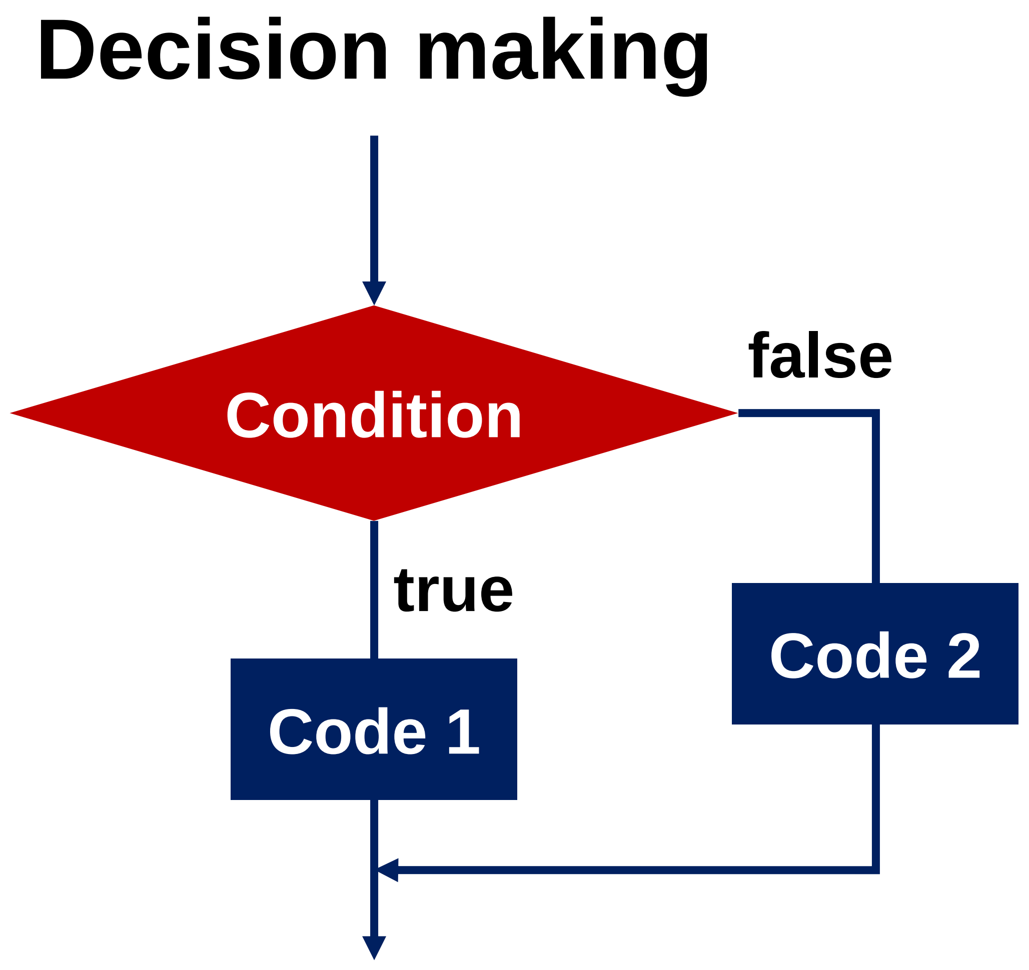
您于前14天内未到达或途经下述城市：

北京市,河北省秦皇岛市,河北省邯郸市... 查看全部



Decision-making in program





If statement

if(condition)	**如果**（条件满足）
{option A}	{A选项}
else	**否则**
{option B}	{B选项}

If statement

Condition
（条件）

Ture/Yes/1
(2<5, 3==3, ’a’!=‘b’)

False/No/0
(1>=5, 3<1, ’a’==‘b’)

If-else-elseif

If only

Block 1

```
int a = 3;
if (a > 10)
{
   // …
}
```

If-else

Block 1

Block 2

```
int a = 3
if (a > 10)
{
   // …
}else
{
   // …
}
```

If-elseif

Block 1

Block 2

Block 3

```
int a = 3
if (a == 1)
{
   // …
}elseif(a == 2)
{
   // …
}elseif(a == 3)
{
   // …
}
```

Nested-if

Parallel if

```
     int a = 3;

     if (a > 10)
     {
1.     // …
     }else
     {
2.     // …
     }
```

Nested if

```
          int a = 3, b = 10;

          if (a > 10)
          {
               if (b < 5)
               {
1.    1.1          // …
               }
          }else{
2.             // …
          }
```

If versus ?

If statement

```
#include <stdio.h>

main ()
{
  int a = 5, b = 10;

  if(a < b)
  {
    printf("b is larger!");
    printf("b is %d", b);
    b++;
    //…
  }
}
```

More space to do things!

? statement

```
#include <stdio.h>

main ()
{
  int a = 5, b = 10;

  int max = a < b ? b : a;
  printf("max is %d", max);
}
```

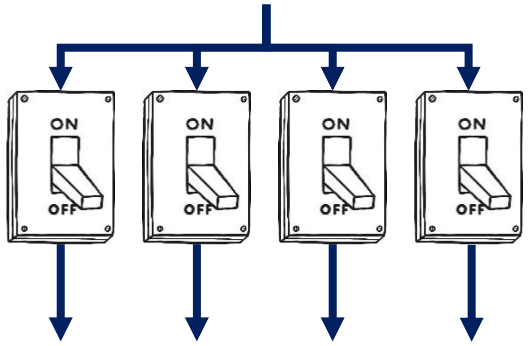
Can only set one variable!

If versus switch

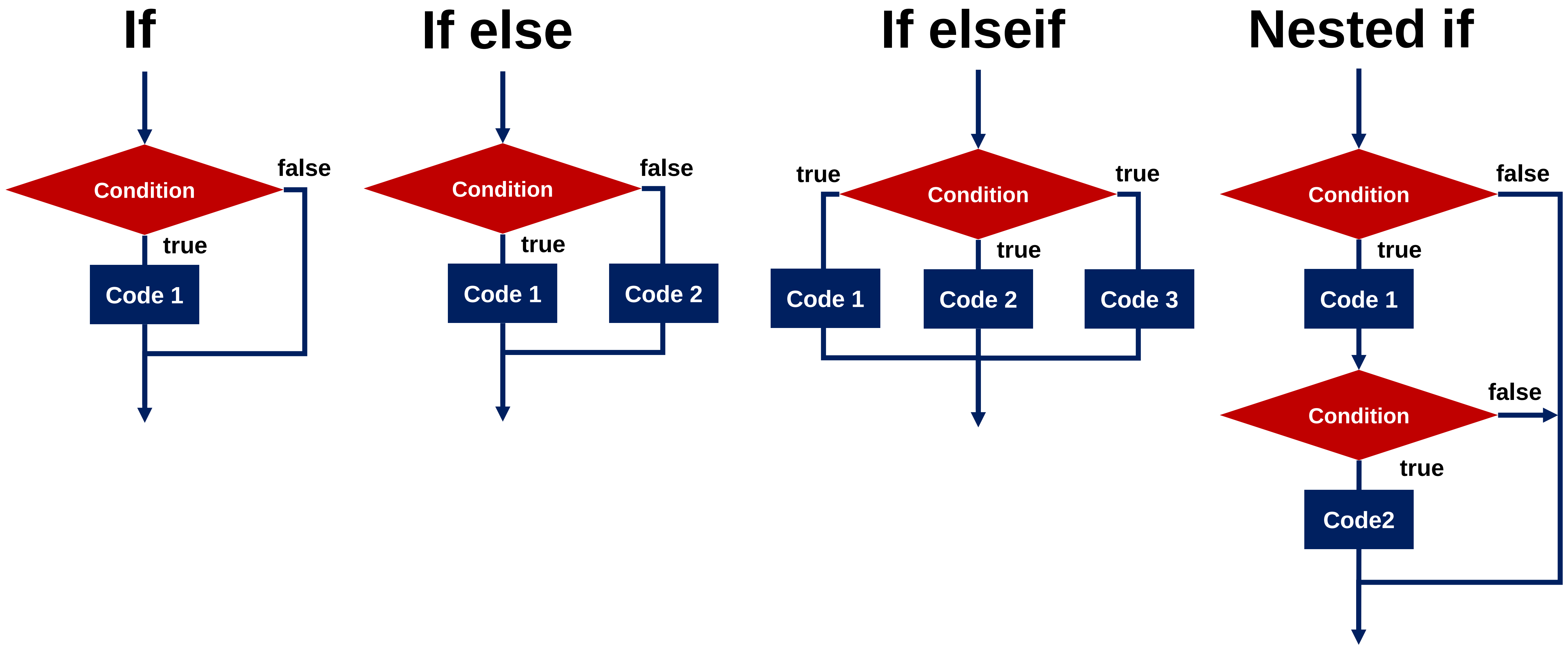
```
int a = 3;

if (a == 1)
{
     // …
}
ifelse(a == 2)
{
     // …
}else{
     // …
}
```

```
int a = 3;

switch(a)
{
  case 1:
     // …
     break;
  case 2:
     // …
     break;
  default:
     // …
}
```

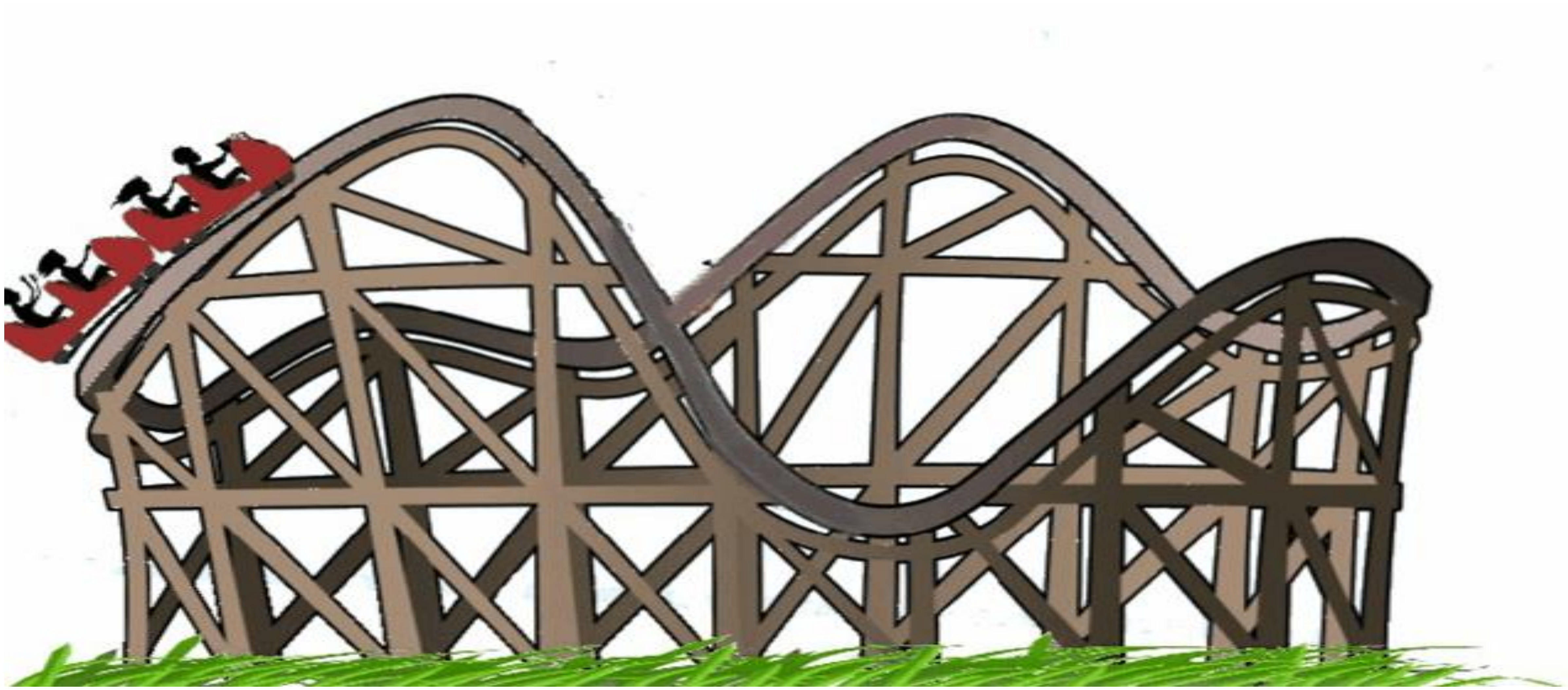
Switch can only express equality!!!



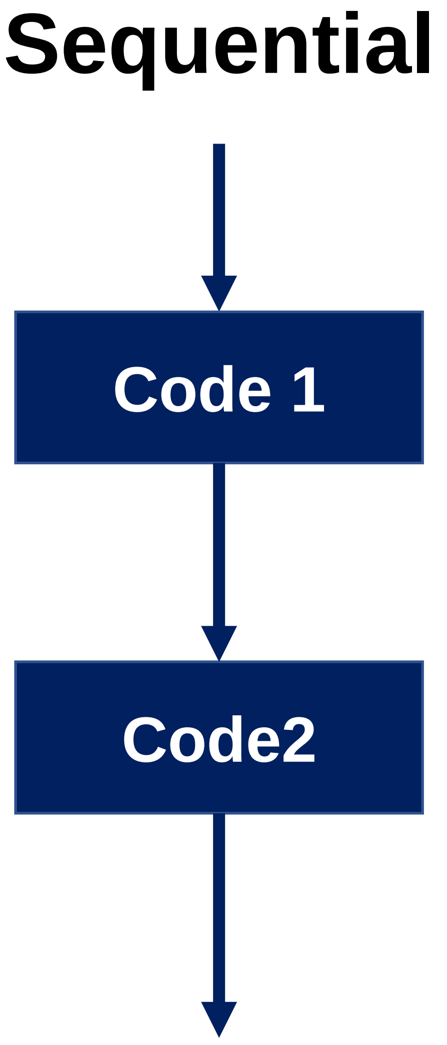
Overview of decision-making

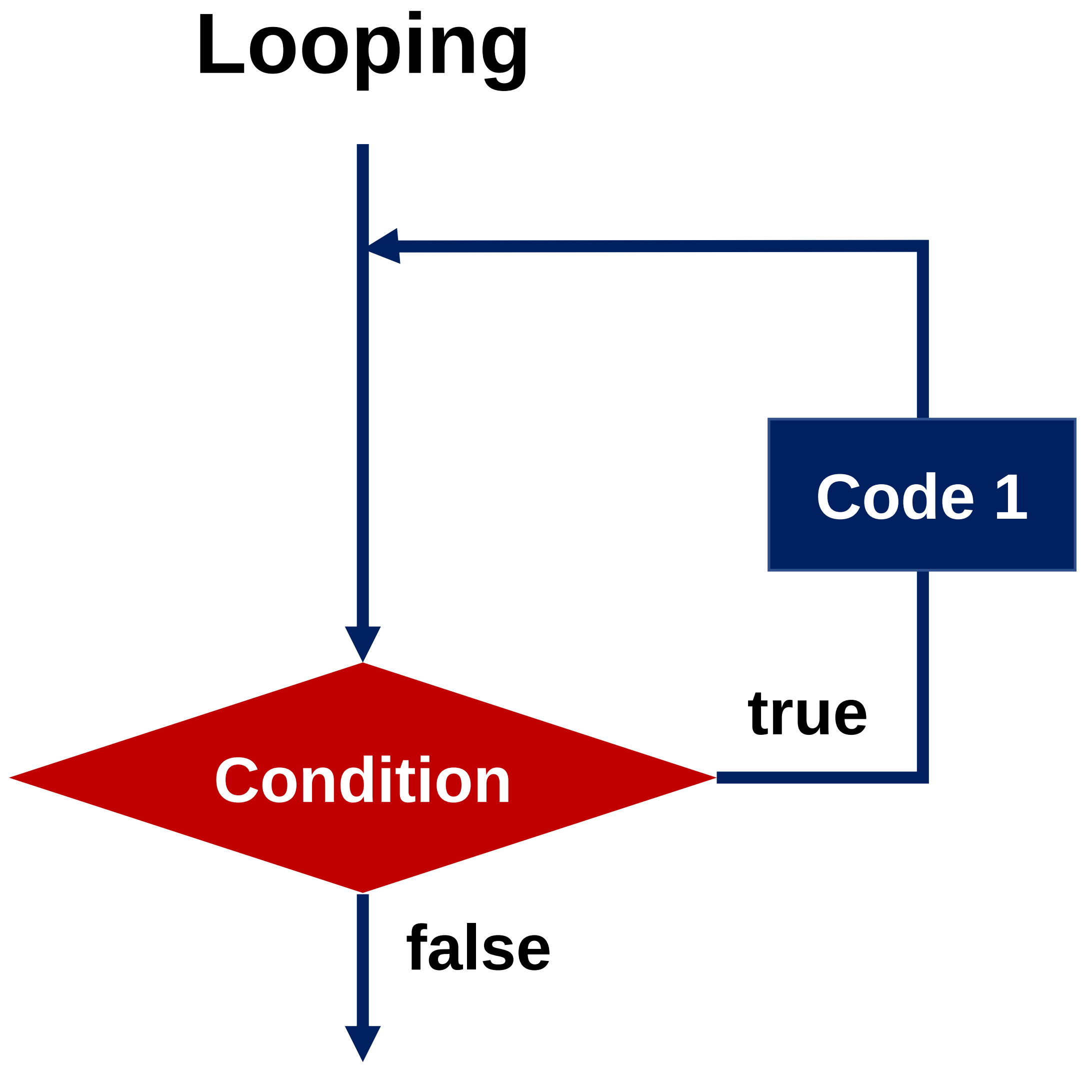


Looping in life

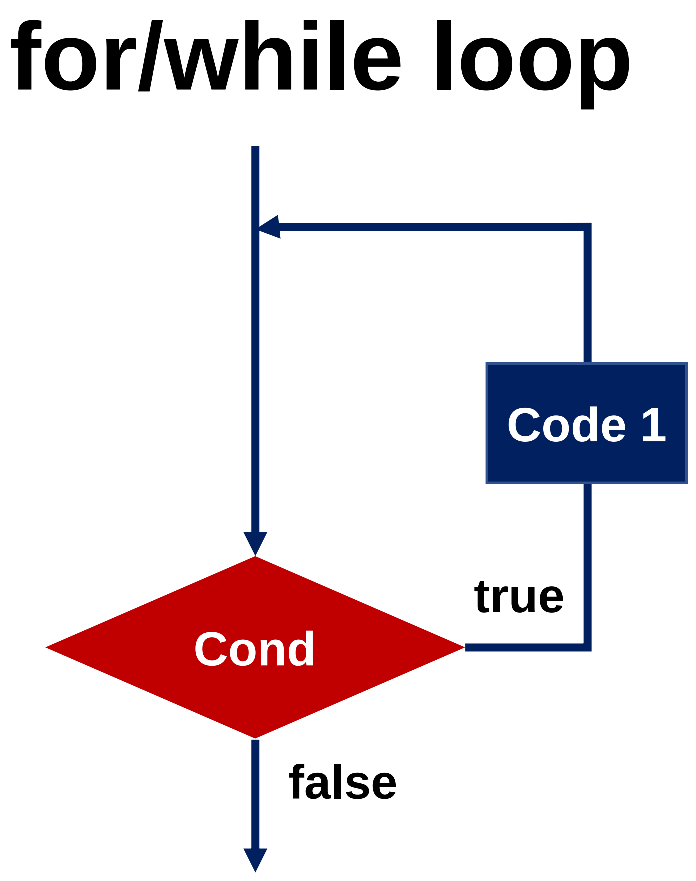


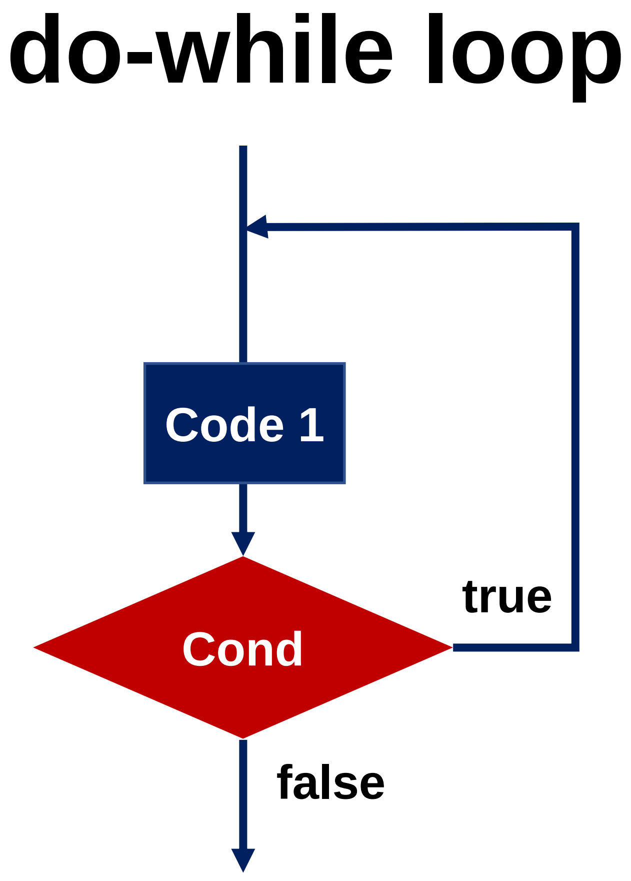
Looping in program

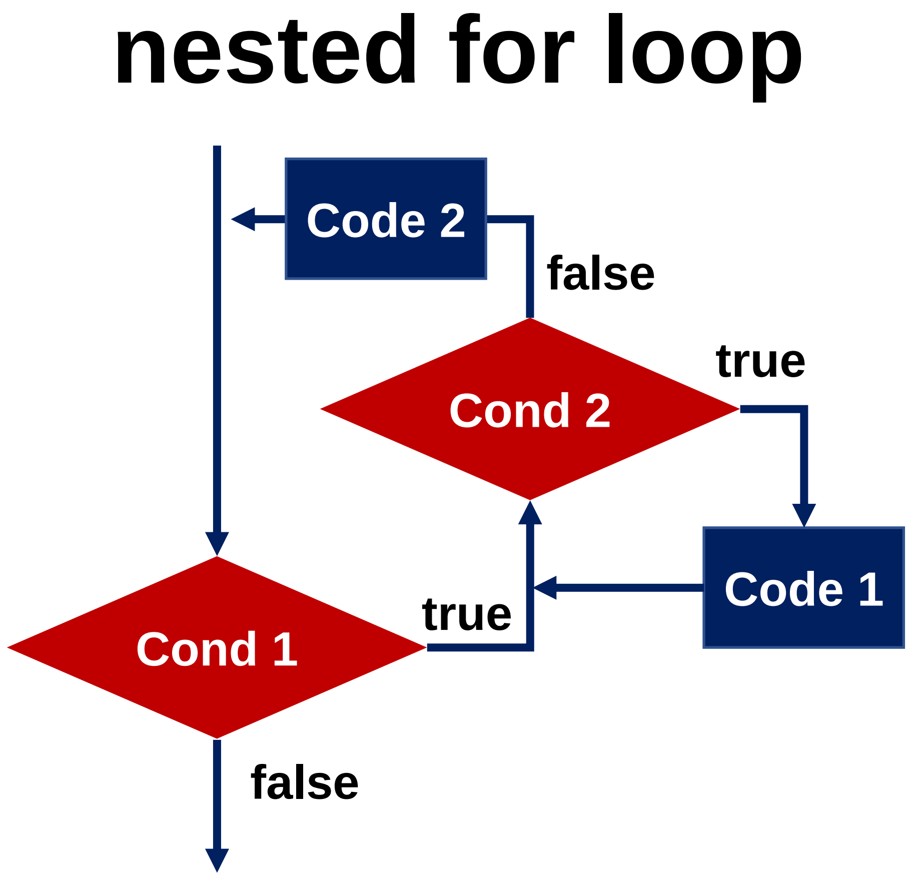




Basic loops in C







For loop

For loop is a control structure that allows repeating the same operation (but different input values) for a specific number of times.

初始化计数器
int a = 0;

计数终止条件
a < 100;

计数
a++

为了

```
for ( init; condition; (de-)increment )
{
    statement;
}
```

For loop

```
for(int a = 0; a < 10; a++)
{
    // …
}
```

increment

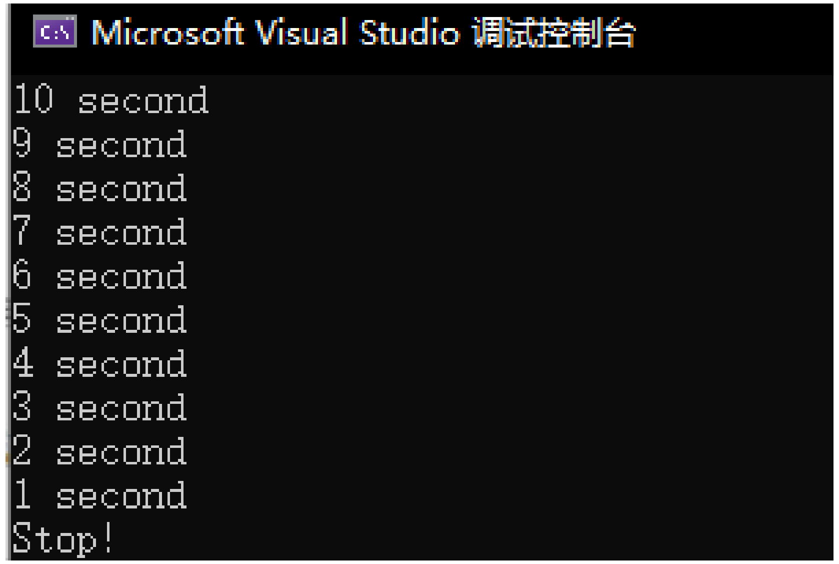
```
for(int a = 100; a >= 0; a--)
{
    // …
}
```

decrement

For loop

Case: create a counter.

```
#include <stdio.h>

main ()
{
    for(int sec = 10; sec>0; sec--)
    {
     printf("%d second\n", sec);
    }
    printf(“Stop!”);
}
```



```
10 second
9 second
8 second
7 second
6 second
5 second
4 second
3 second
2 second
1 second
Stop!
```

While loop

While loop repeatedly executes a statement as long as the condition is true.

初始化计数器
int a = 0;

计数终止条件
a < 100

当

计数
a++;

```
init;

while(condition)
{
     statement;
     (de-)increment;
}
```

While loop

当

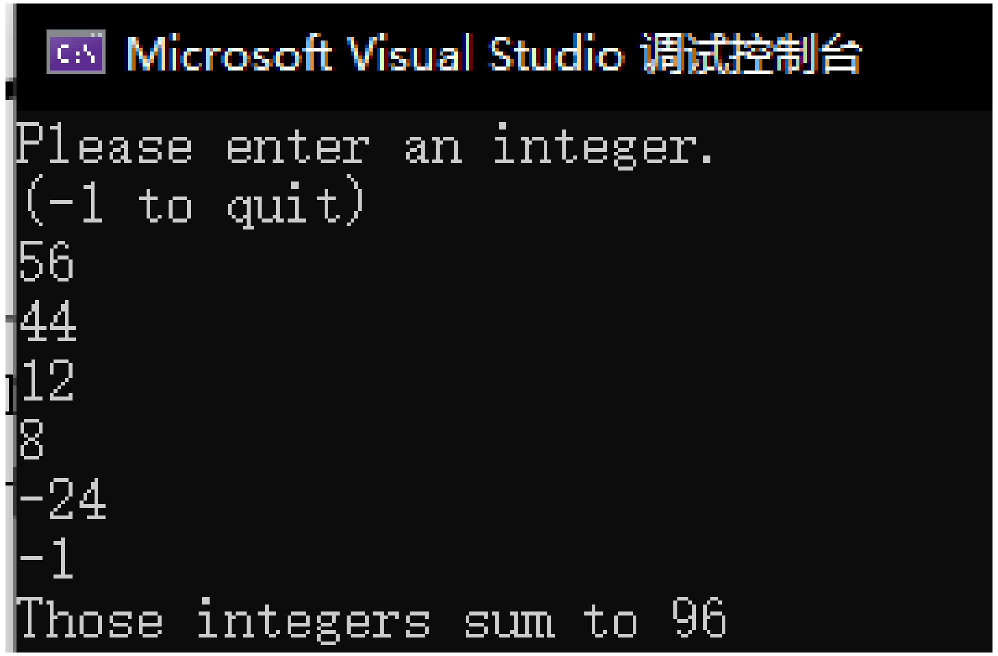
```
int a = 0;
while(a < 10)
{
    // …
    a++;（自增）
}
```

```
int a = 100;
while(a >= 0)
{
    // …
    a--;（自减）
}
```

While loop

Sum the user’s input, exit when input -1.

```
#include <stdio.h>

main ()
{
    printf(“Enter an integer.\n(-1 to quit)\n");
   int input_num = 0;
   int sum = 0;
   while (input_num != -1)
   {
       scanf_s("%d", &input_num);
       sum = sum + input_num;
    }
    printf("Those integers sum to %d", sum);
}
```



```
Please enter an integer.
(-1 to quit)
56
44
12
8
-24
-1
Those integers sum to 96
```

For versus while

```
for(int a = 0; a < 10; a++)
{
    // …
}
```

Same

```
int a = 0;
while(a < 10)
{
    // …
    a++;
}
```

```
for(int a = 100; a >= 0; a--)
{
    // …
}
```

Same

```
int a = 100;
while(a >= 0)
{
    // …
    a--;
}
```

Do-while loop

do-while loop is similar to while loop, it guarantees to execute at least one time.

初始化计数器
int a = 0;

```
init;

do
{
        statement;
        (de-)increment;
} while(condition)
```

做

当

计数
a++;

计数终止条件
a < 100

Do-while loop

```
int a = 0;//100
while(a < 10)
{
    // …
  a++;
}
```

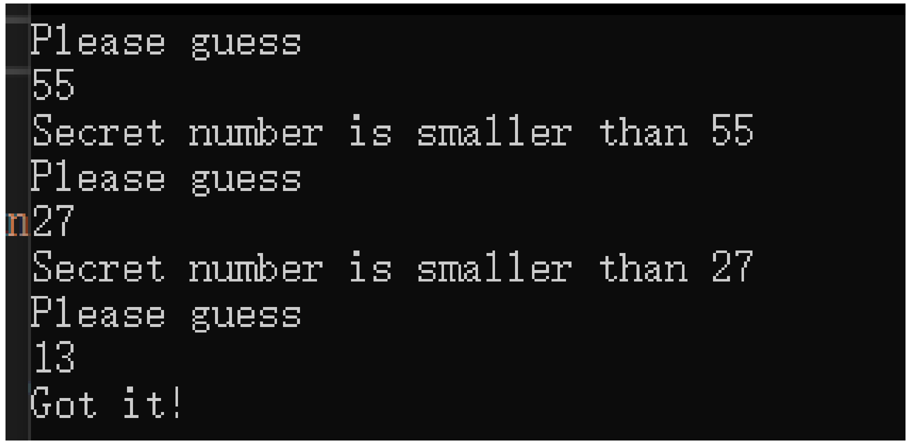
```
int a = 0;//100
do
{
    // …
  a++;
} while(a < 10)
```

When shall we use do-while?

Do-while loop

Case: find the secrete number.

```
#include <stdio.h>
main ()
{
    int num;
    int secret_num = 13;
    do{
      printf("Please guess\n");
      scanf("%d", &num);
      if (num > secret_num) {
          printf("Secret number is smaller than %d\n", num);}
      if (num < secret_num) {
          printf("Secret number is larger than %d\n", num);}
    } while (secret_num != num);
    printf("Got it!\n");
}
```



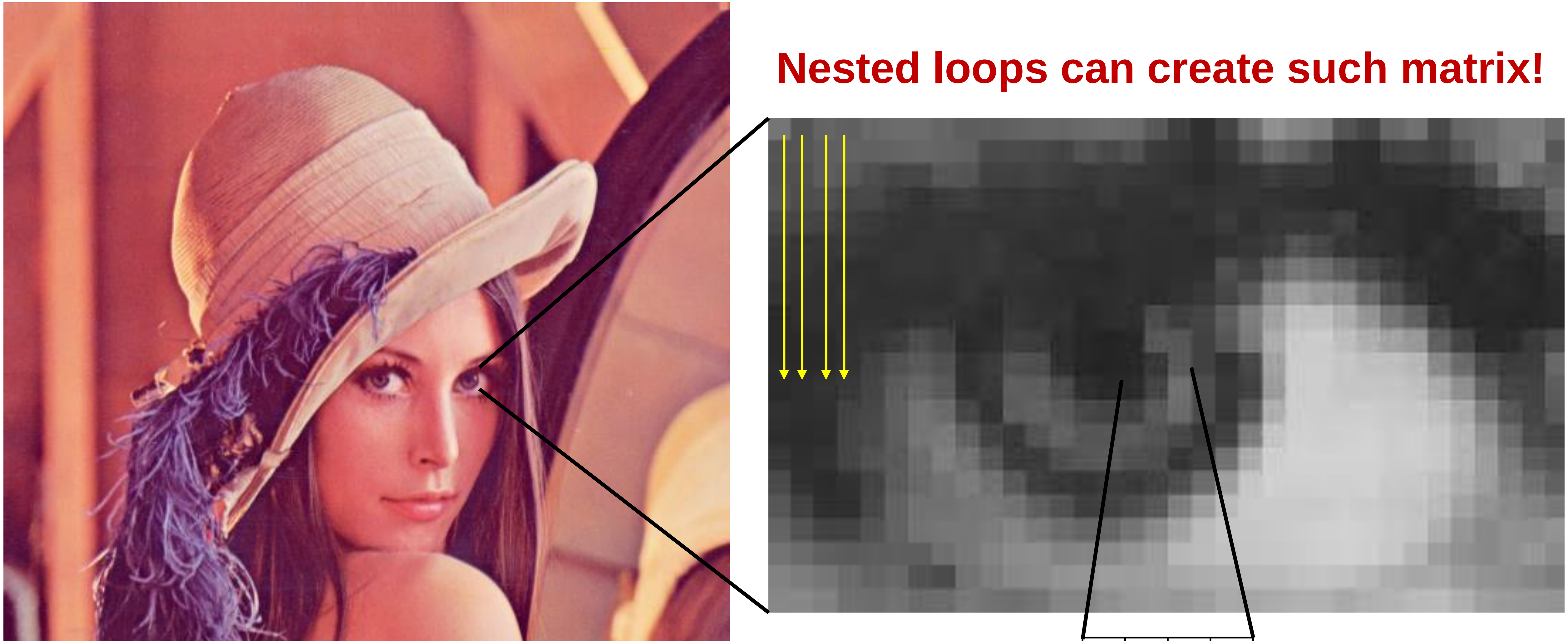
```
Please guess
55
Secret number is smaller than 55
Please guess
27
Secret number is smaller than 27
Please guess
13
Got it!
```

Nested loops

C allows using one loop inside another loop.

```
for ( init; condition; increment )
{
    for ( init; condition; increment )
    {
        // xxxx
    }
}
```

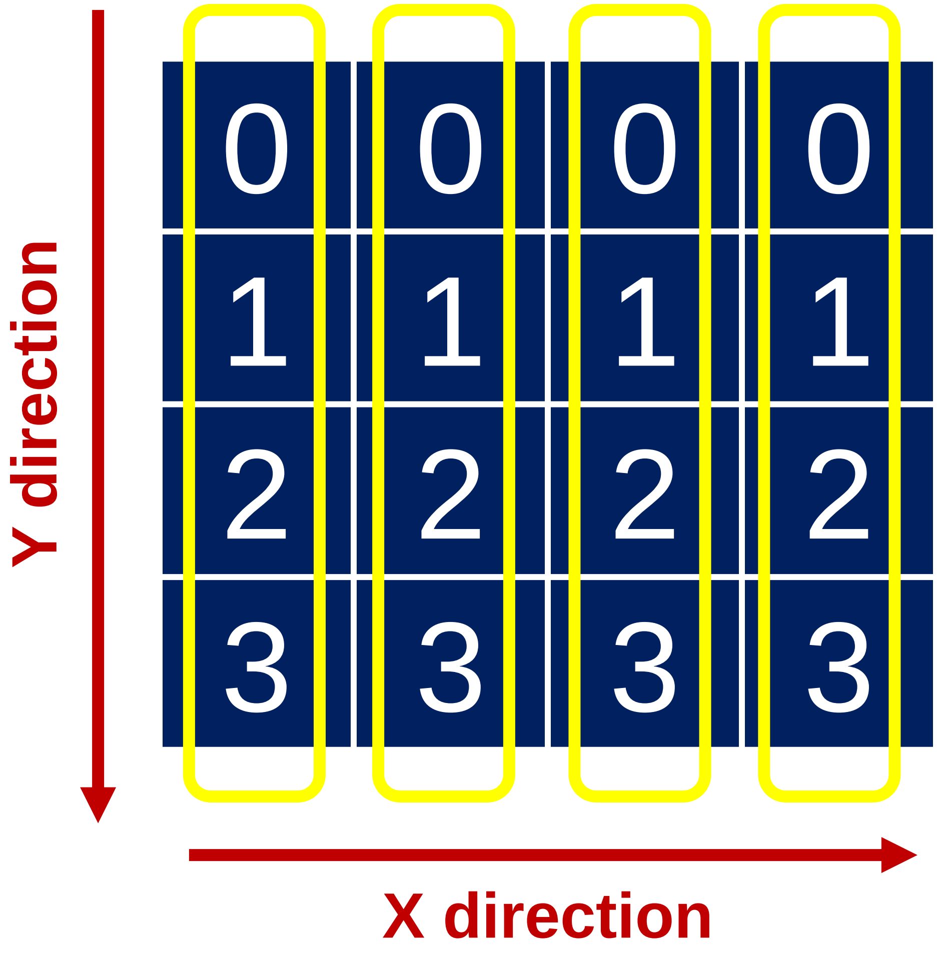
Nested loops



Nested loops can create such matrix!

10	20	20	26
20	41	10	80
60	22	16	84

Nested loops



```
for (int x = 0; x < 4; x++)
{
    for (int y = 0; y < 4; y++)
    {
        // put y at <x, y>
    }
}
```

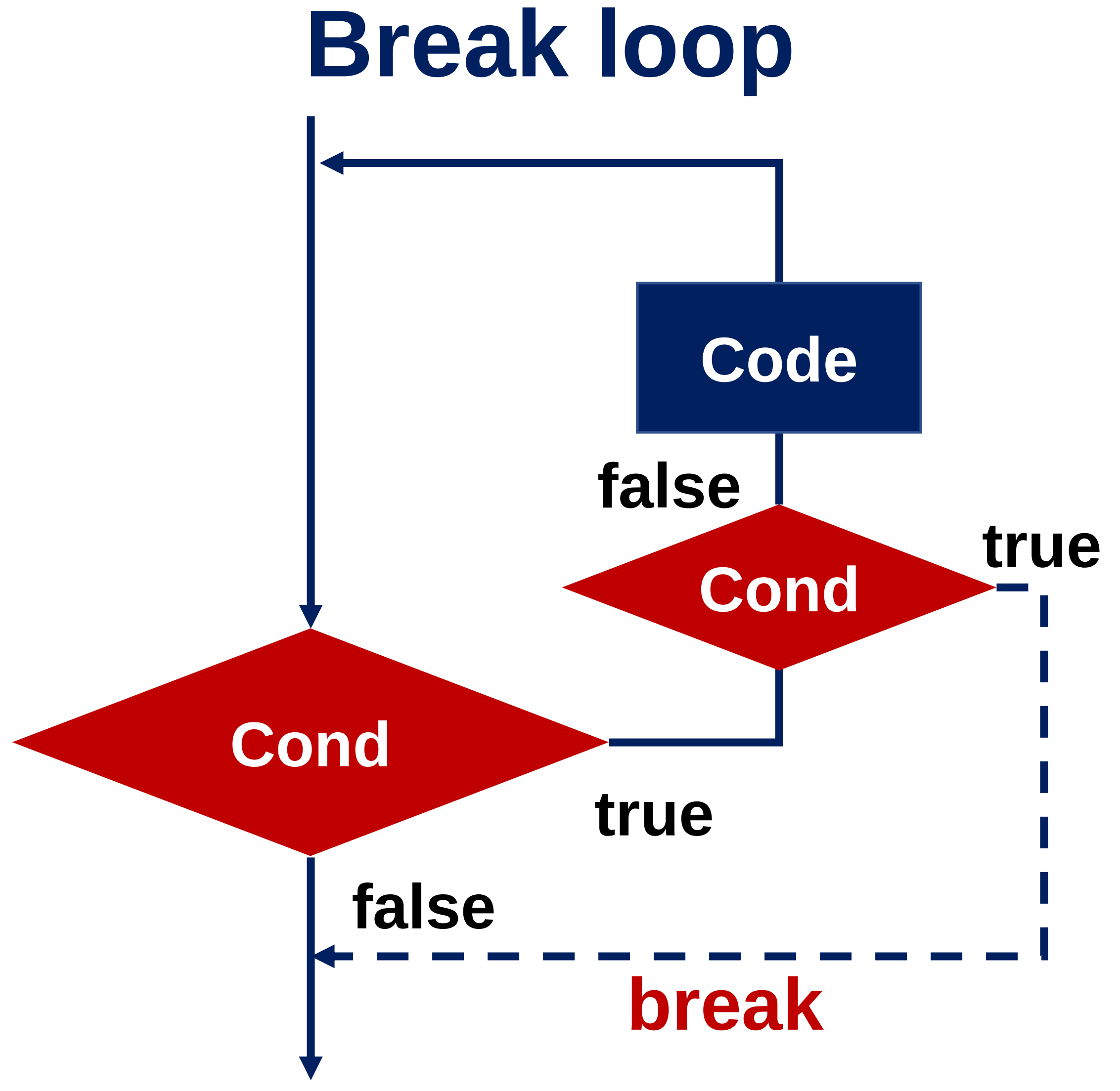
Nested loops

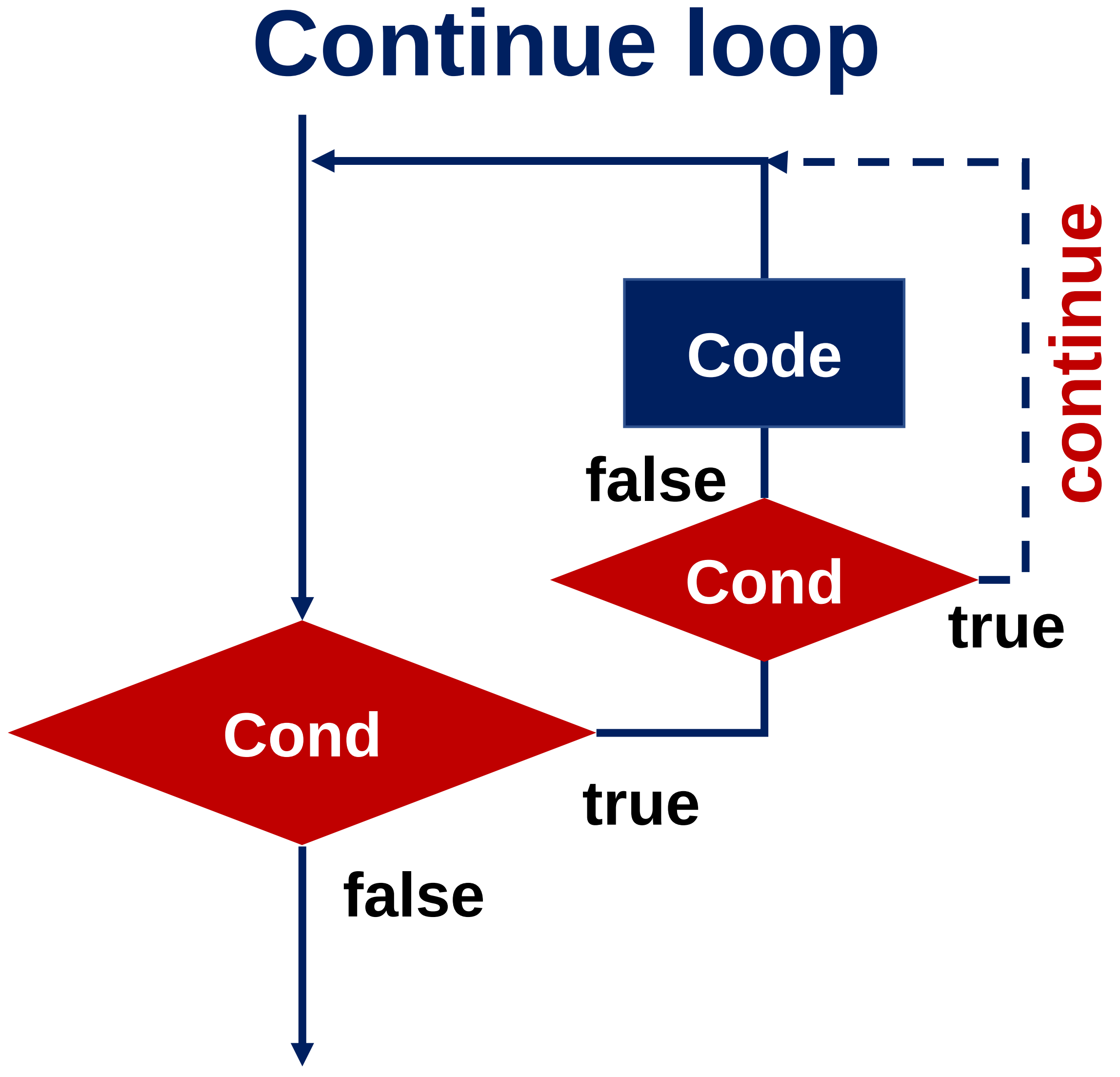
Case: calculate dot product of two matrix

```
float dot_product(int A[], int B[], int rows, int cols)
{
    int* C = (int*)malloc(sizeof(int) * rows * cols);

    for (int i = 0; i < rows; i++){
        for (int j = 0; j < cols; j++){
            C[i * cols + j] = B[i * cols + j] * A[i * cols + j];
        }
    }

    return C;
}
```

$$\begin{matrix} 1 & 2 \\ 3 & 4 \end{matrix} \cdot \begin{matrix} 5 & 6 \\ 7 & 8 \end{matrix} = \begin{matrix} 1\times 5 & 2\times 6 \\ 3\times 7 & 4\times 8 \end{matrix}$$

Break and continue





Break statement

Break terminates the loop when meeting the criterion.

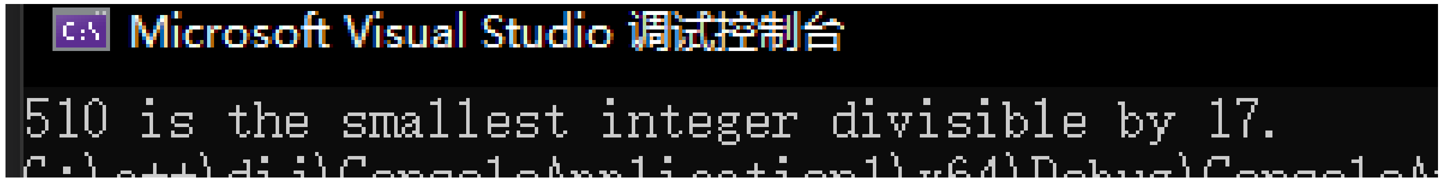
```
for ( init; condition; increment )
{
    if (statement)
        break;
}
```

Break is needed for brute-force searching!

Break statement

Case: output the smallest integer divisible by 17 and larger than 500

```
#include <stdio.h>
int main ()
{
    int num = 500;
    while (1){
      if (num % 17 == 0) {
          printf("%d is the smallest integer divisible by 17.", num);
           break;
      }
      num++;
    }
    return 0;
}
```



Microsoft Visual Studio 调试控制台

```
510 is the smallest integer divisible by 17.
```

Continue statement

Continue forces execution to the next iteration, skipping the code in between.

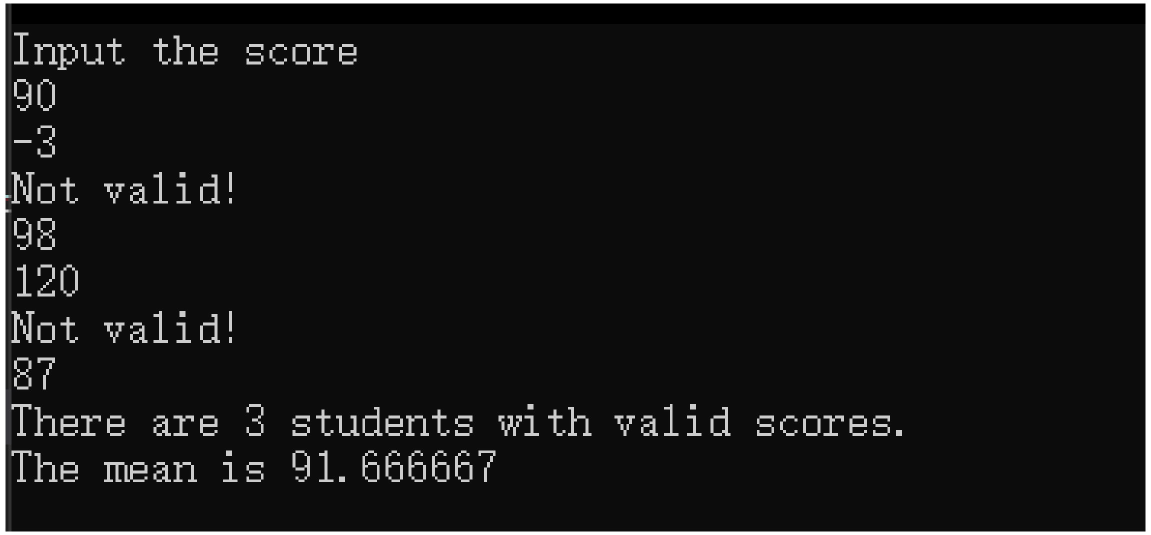
```
for ( init; condition; increment )
{
    if (condition)
        continue;
    // …
}
```

Continue can skip unwanted rounds in looping!

Continue statement

Case: calculate the average score of 5 students with valid scores in [0, 100].

```
#include <stdio.h>
main()
{
    int number = 0, scores = 0,sum =0;
    printf("Input the score\n");
    for (int i = 0;i < 5;i++) {
        scanf ("%d", &scores);
        if (scores < 0 || scores >100) {
            printf("Not valid!\n");
            continue;
        }
        number++; sum += scores;
    }
    printf("There are %d students with valid scores.\nThe mean is %f\n",
number,sum * 1.0 / number);
}
```



```
Input the score
90
-3
Not valid!
98
120
Not valid!
87
There are 3 students with valid scores.
The mean is 91.666667
```

Infinite loop - Virus!

NOTE: A loop becomes **infinite** if a condition never becomes **false**!

```
#include <stdio.h>

int main ()
{
    for( ; ; ) // while(true)
    {
        printf("endless.\n");
    }
    return 0;
}
```

5 Questions

1. What is the difference between **while** and **do-while**?

2. What is the output of below program? ( )

```
int main() {
      char s1[40] = "country", s2[20] = "side";
      int i = 0, j = 0;
      while (s1[i] != '\0')  i++;
      while (s2[j] != '\0')  s1[i++] = s2[j++];
      s1[i] = 0;
      printf("%s\n", s1);
}
```

A. side B. country C. sidetry D. countryside

5 Questions

3. **break** statement can be used to skip one iteration in for loop. Yes/No

4. What is the output of below program?

```
int main() {
        for(int k = 0; k < 5; k++)
        {
                if (k == 3) continue;
                printf("%d ", k);
        }
}
```

5. How to use while loop in C? ( )
   - A. while x < y
   - B. while (x < y)
   - C. if x > y while
   - D. while x < y then

Content

Why do we need array?

```
main()
{
    float student_1;
    float student_2;
    float student_3;
    …
    float student_30;

    scanf("%f", &student_1);
    scanf("%f", &student_2);
    scanf("%f", &student_3);
    …
    scanf("%f", &student_30);
}
```

Array可以批量存储和处理数据！

```
main()
{
    for (int i = 0; i < 30; i++)
    {
        float student_i;
        scanf("%f", &student_i);
    }
}
```

1-D array

C provides a data structure called **array**. It stores a fixed-size collection of elements of the same type.

int array	3	2	1	5	…	8
float array	1.2	4.5	-1.9	3.4	…	8.8
char array	H	R	O	Y	…	P

1-D array

Declare, initialize and access an int array:

- int a[10]; ***// declare***
- a[0] = 3, a[1] = 2, …., a[9] = 7; ***// initialize***

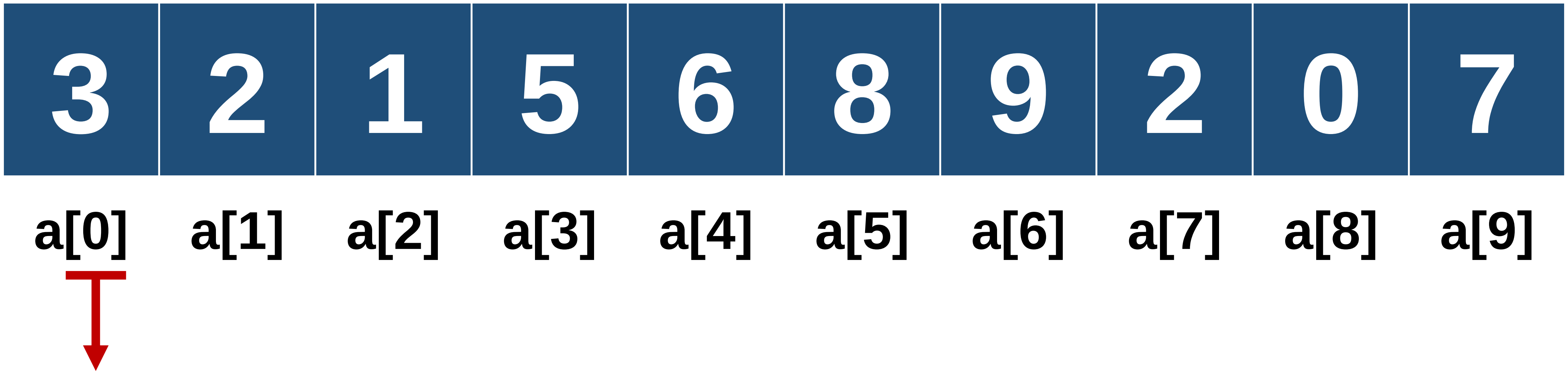
- int a[10] = {3, 2, 1, 5, 6, 8, 9, 2, 0, 7}; ***// declare and initialize***

- int a[] = {3, 2, 1, 5, 6, 8, 9, 2, 0, 7}; ***// declare and initialize***

- printf("a[5] = %d", a[5]); ***// access the array***

1-D array

```
int a[10] = {3, 2, 1, 5, 6, 8, 9, 2, 0, 7}; // length is 10
```

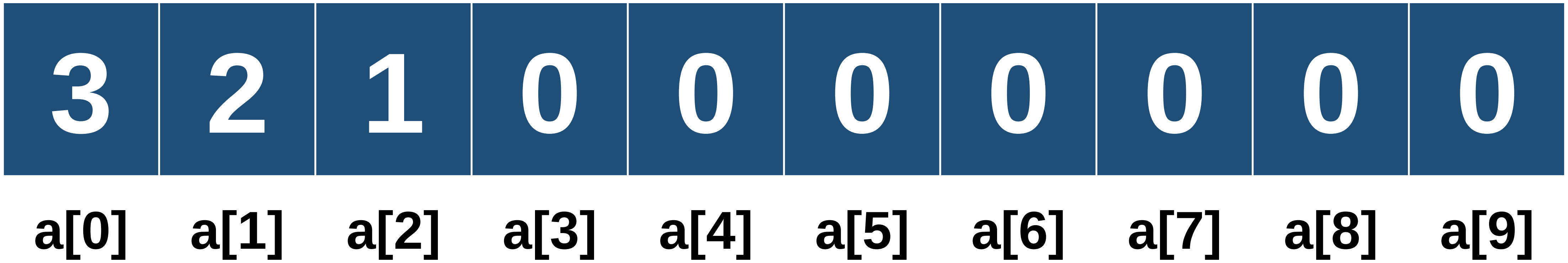


Index starts at 0

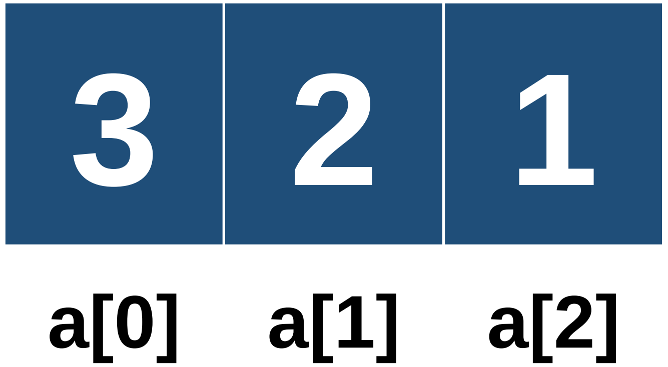
Can we access array by a[10]?

1-D array

```
int a[10] = {3, 2, 1}; // length is 10, fit rests with 0
```



```
int a[] = {3, 2, 1}; // length is 3
```



1-D array

You can also define float array and char array

float array: float a[] = {1.2, -0.6, 1000, -32, 5.34};

1.2	**-0.6**	**1000**	**-32**	**5.34**

char array: char c[] = {‘h’, ‘e’, ‘l’, ‘l’, ‘o’, ‘!’};

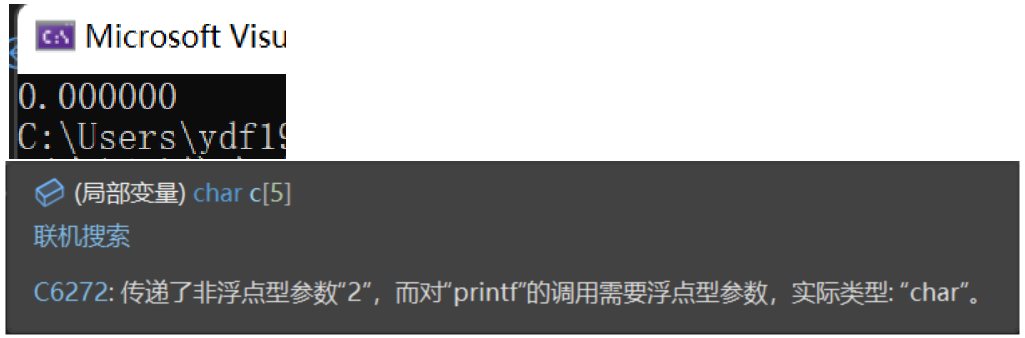
‘h’	**‘e’**	**‘l’**	**‘l’**	**‘o’**	**‘!’**

1-D array

char array: char c[5] = {‘h’, ‘e’, **2, 2.3**, ‘o’}; **// Wrong! Must be in same type!**

int array: int c[5] = {0, 1, 2, **2.5**, 5}; **// Wrong! Must be in same type!**

```
main()
{
    char c[5] = { 'h','e',2,2.3,'o' };
    printf("%f",c[3]);
}
```



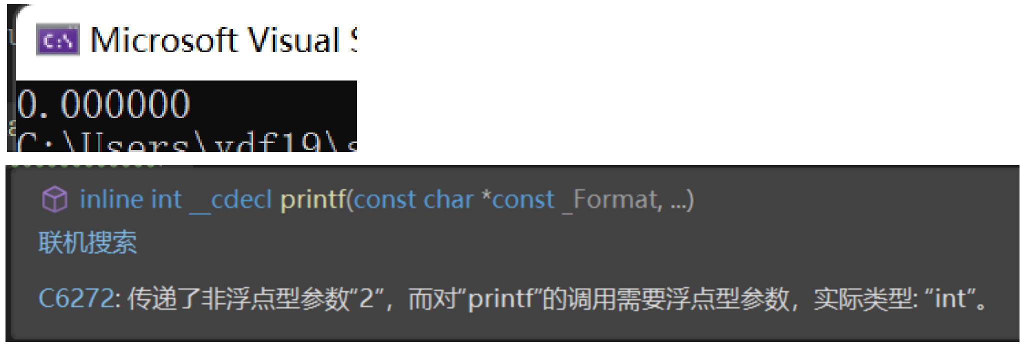
```
0.000000
C:\Users\ydf1
```

(局部变量) char c[5]

联机搜索

C6272: 传递了非浮点型参数“2”，而对“printf”的调用需要浮点型参数，实际类型: “char”。

```
main()
{
    int c[5] = {0,1,2,2.5,5};
    printf("%f",c[3]);
}
```



```
0.000000
C:\Users\ydf19\
```

inline int __cdecl printf(const char *const _Format, ...)

联机搜索

C6272: 传递了非浮点型参数“2”，而对“printf”的调用需要浮点型参数，实际类型: “int”。

2-D array

Declare and initialize a 2D int array

3	2	5
1	7	6

- int a[2][3]; **// 2 rows x 3 columns**
- a[0][0] = 3; a[0][1] = 2; a[0][2] = 5;
- a[1][0] = 1; a[1][1] = 7; a[1][2] = 6;

Access array: printf(“a[1][1] = %d”, a[1][1]);

1	0	0	2
0	1	0	0
0	2	1	4

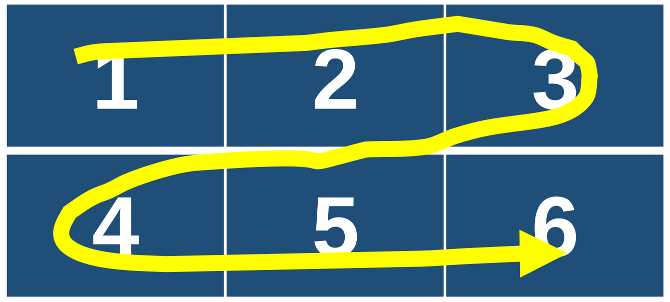
- int a[3][4]; **// 3 rows x 4 columns**
- a[0][0] = 1; a[0][1] = 0; a[0][2] = 0; a[0][3] = 2;
- a[1][0] = 0; a[1][1] = 1; a[1][2] = 0; a[1][3] = 0;
- a[2][0] = 0; a[2][1] = 2; a[2][2] = 1; a[2][3] = 4;

Access array: printf(“a[2][3] = %d”, a[2][3]);

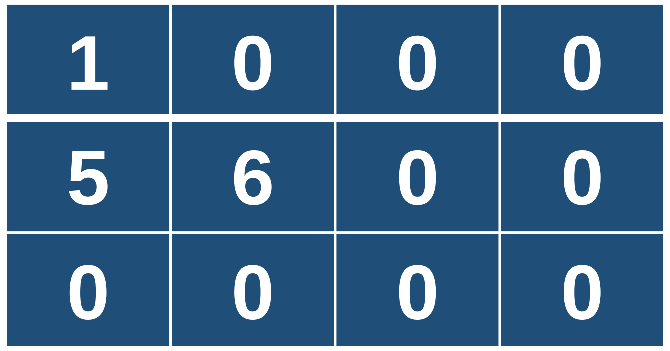
2-D array

Declare and initialize a 2D int array

- int a[2][3] = {{1, 2, 3}, {4, 5, 6}};
- int a[2][3] = {1, 2, 3, 4, 5, 6}; // **preferred!**
- int a[][3] = {1, 2, 3, 4, 5, 6}; // 2 x 3 mat
- int a[3][4] ={{1}, {5, 6}}; // 3 x 4 mat



1	2	3
4	5	6



1	0	0	0
5	6	0	0
0	0	0	0

3-D/N-D array

Declare and initialize a 3-D/N-D int array

- int a[2][3][4];
- a[0][0][0] = 1; a[0][1][2] = 3; a[1][0][3] = 2; // **preferred!**
- int a[2][3][4]= {{{1, 2, 3}, {4, 5, 6}},{{2, 4, 5}, {2, 4, 2}}…};

- int a[2][3][4][2];
- a[0][0][0][0] = 1; a[0][1][2][0] = 3; a[1][0][3][1] = 2;

Use for loop to define 2D/3D array

2D array

```
int n[4][5];
for (int x = 0; x < 4; x++)
{
    for (int y = 0; y < 5; y++)
    {
        n[x][y] = x+y;
    }
}
```

3D array

```
int n[2][2][3];
for (int x = 0; x < 2; x++)
{
    for (int y = 0; y < 2; y++)
    {
        for (int z = 0; z < 3; z++)
        {
            n[x][y][z] = x+y+z;
        }
    }
}
```

String

String is an array of characters.

```
char c[10] = {‘I’, ‘ ’, ‘a’, ‘m’, ‘ ’, ‘h’, ‘a’, ‘p’ , ‘p’, ‘y’}; // length is 10
char c[10] = {“I am happy”};
char c[] = {“I am happy”};
char c[] = “I am happy”; // preferred
```

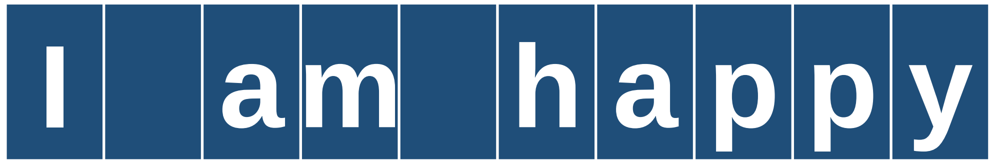
I		a	m		h	a	p	p	y
c[0]	**c[1]**	**c[2]**	**c[3]**	**c[4]**	**c[5]**	**c[6]**	**c[7]**	**c[8]**	**c[9]**

1D and 2D String

1D char array holds the characters!

char c[10] = “I am happy”;

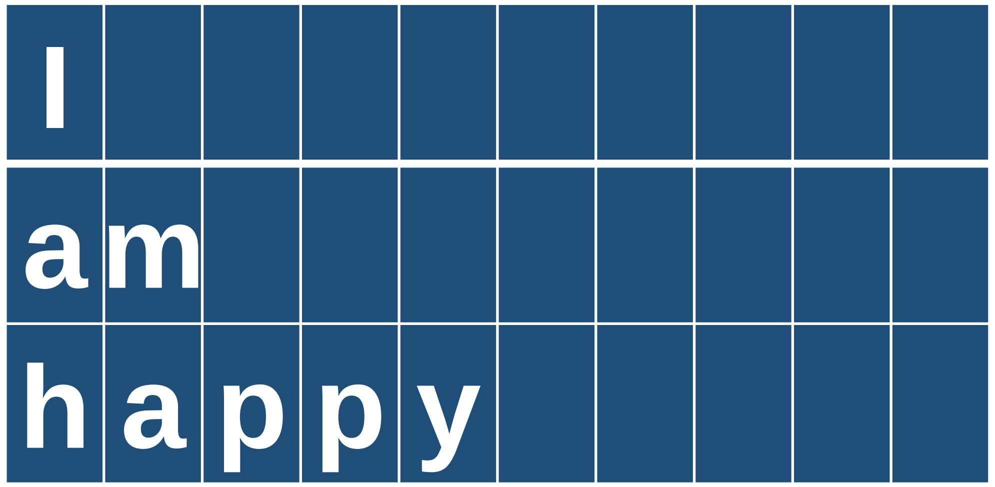
Machine thinks it as a single “word”!



2D char array holds the words!

char c[3][10] = {“I”, “am”, “happy”};

Machine thinks it as a group of word!



String

```
char c[10] = {‘S’, ‘ U’, ‘S’, ‘T’, ‘e ’, ‘c’, ‘h’}; // length is 10
char c[10] = {“SUSTech”};
char c[] = {“SUSTech”};
char c[] = “SUSTech”; // preferred
```

S	U	S	T	e	c	h	\0	\0	\0
c[0]	c[1]	c[2]	c[3]	c[4]	c[5]	c[6]	c[7]	c[8]	c[9]

String operations

C supports a wide range of functions that manipulate strings.

Operators	**Description**	**Example s1=A, S2 = B;**
strcpy(s1, s2)	Copy s2 into s1	s1 = B
strcat(s1, s2)	Concatenate s1 and s2	S1 = AB
strlen(s1)	Return length of s2	Length = 1
strcmp(s1, s2)	Compare s1 and s2	A<B, return -1
strlwr(s1)	Convert s1 to lower case	A to a
strupr(s1)	Convert s1 to upper case	A to A

strcpy(s1, s2)

```
char str1[12] = "Hello";
char str2[12] = "World";
char str3[12];

strcpy(str3, str1);
printf("str3 = %s\n", str3); //Hello

strcpy(str3, str2);
printf("str3 = %s\n", str3); //World
```

strcat(s1, s2)

```c
char str1[12] = "Hello";
char str2[12] = "World";
char str3[12] = "123";

strcat(str1, str2);
printf("str1 = %s\n", str1); //HelloWorld

strcat(str3, str2);
printf("str3 = %s\n", str3); //123World
```

strlen(s1)

```
char str1[12] = "Hello";
char str2[] = "World";
char str3[12];

printf("str1 = %s\n", strlen(str1)); //5

printf(“str2 = %s\n”, strlen(str2)); //5

printf("str3 = %s\n", strlen(str3)); //0
```

sizeof(s1)

```
char str1[12] = "Hello";
char str2[] = "World";
char str3[12];

printf("str1 = %s\n", sizeof(str1)); //12

printf(“str2 = %s\n”, sizeof(str2)); //6, end with ‘\0’

printf("str3 = %s\n", sizeof(str3)); //12
```

strcmp(s1, s2)

```
char str1[] = "ABCD";
char str2[] = "BCD";
char str3[] = "ABCE";
char str4[] = "1234";
```

```
str1 > str2 →  1
str1 < str2 → -1
str1 = str2 →  0
```

```
printf("cmp = %d\n", strcmp(str1, str2)); //-1

printf("cmp = %d\n", strcmp(str1, str3)); //-1

printf("cmp = %d\n", strcmp(str1, str1)); //0
```

strlwr(s1)

```
char str1[] = "ABCD";
char str2[] = "abcd";
char str3[] = “012abcDE";

printf("strlwr = %d\n", strlwr(str1)); //abcd

printf("strlwr = %d\n", strlwr(str2)); //abcd

printf("strlwr = %d\n", strlwr(str3)); //012abcde
```

strupr(s1)

```
char str1[] = "ABCD";
char str2[] = "abcd";
char str3[] = “012abcDE";

printf("strupr = %d\n", strupr(str1)); //ABCD

printf("strupr = %d\n", strupr(str2)); //ABCD

printf("strupr = %d\n", strupr(str3)); //012ABCDE
```

Summary

- We can use **array** to hold many data for group processing
- Array has the **fixed size** and can only be used to hold data with **same type**
- **Different types of array** can be created, e.g. int array, float array, char array (string)
- **Different dimensional array** can be created, from 1D array to ND array
- Array enables the processing of **vectors, matrices, strings**, etc.

5 Questions

1. If we use the array name as the argument for the function, what does the argument stand for? ( )

A. The value of the first element in the array
B. The value of all elements in the array
C. The address of the first element in the array
D. The address of all elements in the array

2. How to declare a 1D array? ( )

A. int a(10);   B. int a{10};   C. int [10]a;   D. int a[10];

3. How to declare a 2D array? ( )

A. int a[3][];   B. float a(3,4);   C. double a[3][4];   D. float a(3)(4);

5 Questions

4. Which statement can initialize a 2D array correctly? ( )

A. int a[2][3]={{1,2},{3,4},{5,6}};
B. int a[2][3]={{1,2},{},{4,5}};
C. int a[][3]={1,2,3,4,5,6};
D. int a[2][]={{1,2},{3,4},{4,5}};

5. Which statement is correct in checking if string s1 equals to string s2? ( )

A. if(s1 == s2)   B. if(s1 = s2)   C. if(strcpy(s1,s2))   D. if(strcmp(s1,s2)==0)

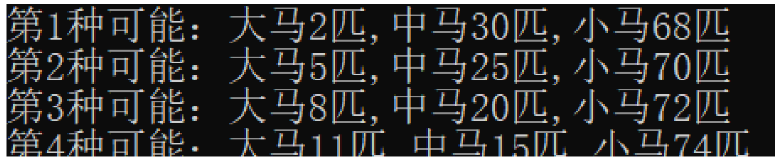
Assignment

1. Write a program that can convert a decimal integer to a binary number and print the binary number

a) You need to consider the case that this integer is negative
b) Use scanf to enter the number
c) Test input 34, -39

2. There are 100 horses in total, they can carry 100 bags of goods, a big horse can carry 3 bags; a medium horse can carry 2 bags, and two small horses can carry one bag. How many big horses, medium horse and small horses are in the total 100 horses? Write a program to calculate and print the number of big horses, medium horses and small horses separately.

a) The number of horses mast be a positive integer
b) You can refer to “chicken and rabbit in the same cage”
c) The form of the answer is as follows



```
第1种可能: 大马2匹,中马30匹,小马68匹
第2种可能: 大马5匹,中马25匹,小马70匹
第3种可能: 大马8匹,中马20匹,小马72匹
第4种可能: 大马11匹,中马15匹,小马74匹
```

Assignment

3.Write a function to calculate the sum of all the elements on the boundary of a 2-D matrix.

a) All the elements on the boundary of below 2-D matrix is denoted in blue

$$\begin{matrix} 3 & 0 & 0 & 3 \\ 2 & 5 & 7 & 3 \\ 1 & 0 & 4 & 2 \end{matrix}$$

b) You can use 1-D array as the arguments of the function(use 1-D array to represent 2-D matrix)

c) Test input

$$\begin{matrix} 5 & 0 & 6 & 3 \\ 2 & 8 & 7 & 4 \\ 1 & 9 & 4 & 3 \end{matrix}$$