Introduction to C Programming

Lecture 2: basics

Wenjin Wang
wangwangj3@sustech.edu.cn

9-16-2022

About the course

- **Lecturer**: Wenjin Wang (Tom), associate professor of BME
- **Lecture time/location**: Friday 7-8 （三教**107**）, 9-10（三教**508, 509**机房）
- **Office:** 工学院南楼 **637** “无线健康感知” 实验室
- **Email**: wangwj3@sustech.edu.cn
- **Assistance**:
  - Dongfang Yu (Frank)
  - Dan Li
  - Tingdan Luo

22级生医工系本科生咨询群

该二维码7天内(9月16日前)有效，重新进入将更新

Grading

- **Final exam: 40%**
  (you will find most answers in the slides and assignments)

- **Assignments: 50%**
  (5 +1 programming tasks released per lecture, try to finish in the lab)

- **Course/lab attendance: 10%**
  (easiest way to earn credits!)

Policies

- Lecture will be in English (with Chinese), try to practice your both languages (C and English) in the class
- Slides will be released on Thursday (before lecture)
- Assignment needs to be submitted on Monday, needs to be finished individually
- Assignment will be reviewed in the lab session a week later

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	Memory control & file I/O	2022.10.28

Nr.	Lecture	Date
9	Head files & pre-processors	2022.11.4
10	Review of lectures	2022.11.11
11	Soul of programming: Algorithms I	2022.11.25
12	Soul of programming: Algorithms II	2022.12.2
13	R&D project	2022.12.9
14	R&D project	2022.12.16
15	R&D project	2022.12.23
16	Summary	2023.12.30

Recap last lecture

- Machine and machine intelligence are everywhere. The way to control machine is by **programming**.
- C is a high-level language that is comfortable to use while still efficient for machines. It is **popular and ubiquitous in industry**, especially for edge devices.
- **C + AI** (or domain knowledge) makes you different. Best way to learn is **practice**!!!
- You already know how to write the first “HelloWorld” program.

Recap last lecture

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;
}
```

Recap last lecture

Exit

Entrance

Do something!

```
#include<stdio.h>

int main(int a)

{

    printf("Hello World %d!", a);

    return 0;

}
```

Recap last lecture

```
printf(format, arguments);
```

```
printf(“Hello World”);
printf(“Hello World\n”);
printf(“my ID is %d”, 3);
printf(“my name is %c”, ‘A’);
```

Objective of this lecture

You can use C to make basic calculations with I/O!

Content

1. Bit and byte
2. Data types and variables
3. Operations
4. I/O

Content

Bit and byte

Bit (位)

The smallest unit for storage (atomic),

0 or 1

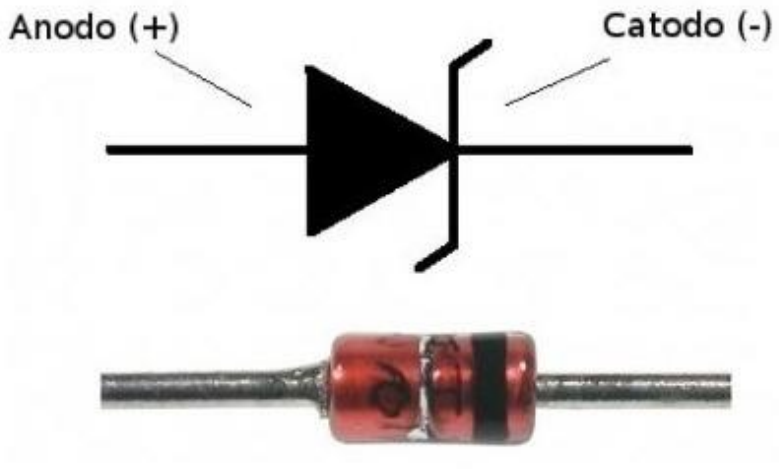
Byte (字节)

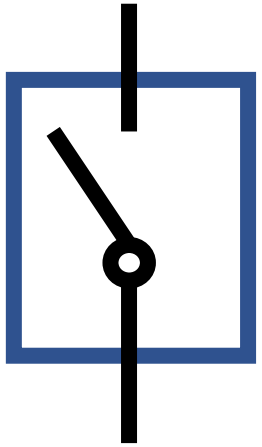
The smallest unit for information storage,

1 byte = 8 bits

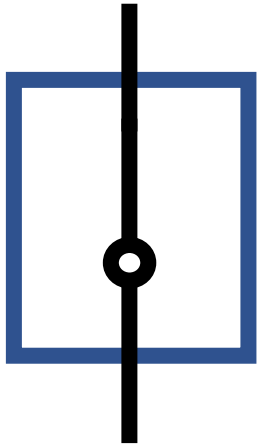
Bit

Computer is nothing but a vast collection of **diodes (on and off)**, denoting the state of 0 and 1.





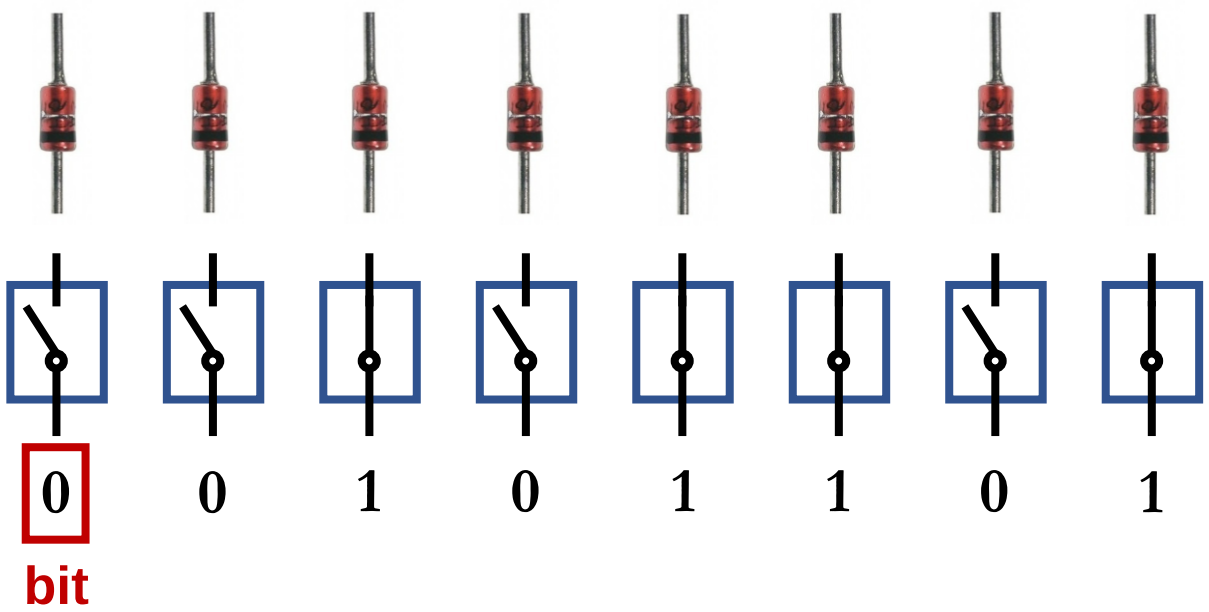
Off
“0”
False/No



On
“1”
True/Yes

Byte

1 byte = 8 bits



More diodes = More bits

More complex information

- 1024 bytes = 1 KB (Kilobyte)
- 1024 KB = 1 MB (Megabyte)
- 1024 MB = 1 GB (Gigabyte)
- 1024 GB = 1 TB (Terabyte)
- 1024 TB = 1 PB (Petabyte)

1 KB = 1024 (2^10) bytes = 8192 bits

Decimal numbering system

2 0 2 2

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

Use 10 as basis

Decimal numbering system

3 8 4 6

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

Use 10 as basis

Binary numbering system

1 0 1 1

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

$$= 11$$

Use 2 as basis

Binary numbering system

1 0 0 1 0

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

$$= 18$$

Use 2 as basis

Decimal to binary

2	2022	 0
2	1011	 1
2	505	 1
2	252	 0
2	126	 0
2	63	 1
2	31	 1
2	15	 1
2	7	 1
2	3	 1
2	1	 1
	0	

2022

1 1 1 1 1 1 0 0 1 1 0

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	

3846

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

(float) Decimal to binary

88.8125 →

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

88

1011000

1101

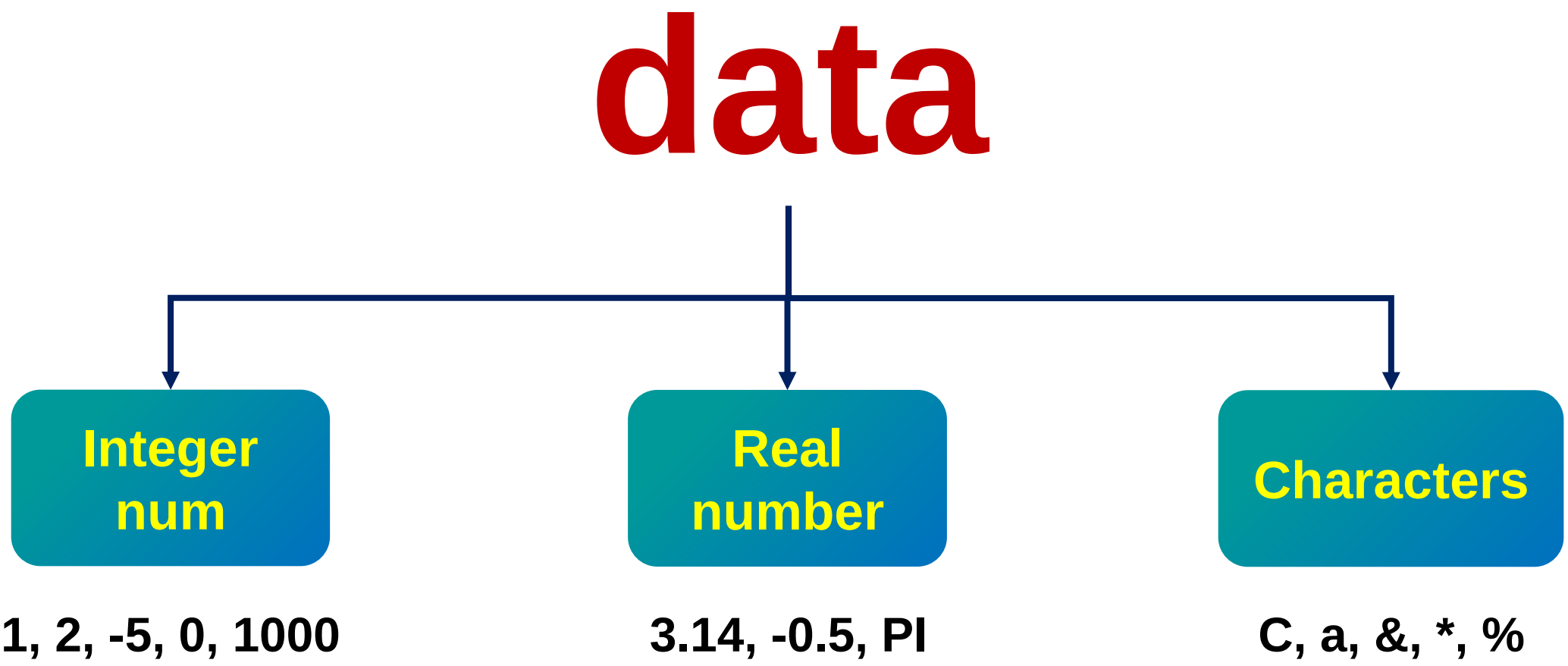
1011000.1101

Binary to (float) decimal

1 0 1 1 0 0 0 . 1 1 0 1

$$= (1 * 2^{6}) + (0 * 2^{5}) + (1 * 2^{4}) + (1 * 2^{3})$$
$$+ (1 * 2^{0}) + (0 * 2^{2}) + (0 * 2^{1}) + (0 * 2^{0})$$
$$+ (1 * 2^{-1}) + (1 * 2^{-2}) + (0 * 2^{-3}) + (1 * 2^{-4})$$
$$= 88.8125$$

Use byte to store data



Use byte to store integer number

0	1	0	0	1	0	1	1

$$2^7 - 1$$

Sign

0 → -

1 → +

1 byte denotes range [-128, 127]

Use byte to store real number

Use 2 as basis

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

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

+	110	.0110001101
Sign	Index	Values
1 bit	8 bits	23 bits

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]

Bit and byte

- ✓ Bit is the atomic unit for machine (0 and 1)
- ✓ Byte is the smallest unit for information storage, 1 byte = 8 bits
- ✓ A collection of bytes can be used to denote integer number, real number, characters
- ✓ Machine interprets everything in the binary format

Content

1. Bit and byte
2. Data types and variables (tingdan)
3. Operations
4. I/O

Data types and variables

You may still remember HelloWorld example?

int is a data type, ask the program to return an integer number!

Your first C program

```
#include <stdio.h>

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

    return 0;
}
```

Data types and variables in life


```
int firing_rate;
float load_speed;
int move_speed;
float switch_speed;
int damage;
int capacity;
```

Data types and variables in life

Name in home country:	Kylian Mbappé Lottin
Date of birth:	Dec 20, 1998
Place of birth:	Paris
Age:	23
Height:	1,78 m
Citizenship:	France
Position:	attack - Centre-Forward
Foot:	right
Player agent:	Relatives
Current club:	Paris Saint-Germain
Joined:	Jul 1, 2018
Contract expires:	Jun 30, 2025
Date of last contract extension:	May 21, 2022
Outfitter:	Nike

Main position

Main position:
Centre-Forward

Other position:
Left Winger
Right Winger

Market value

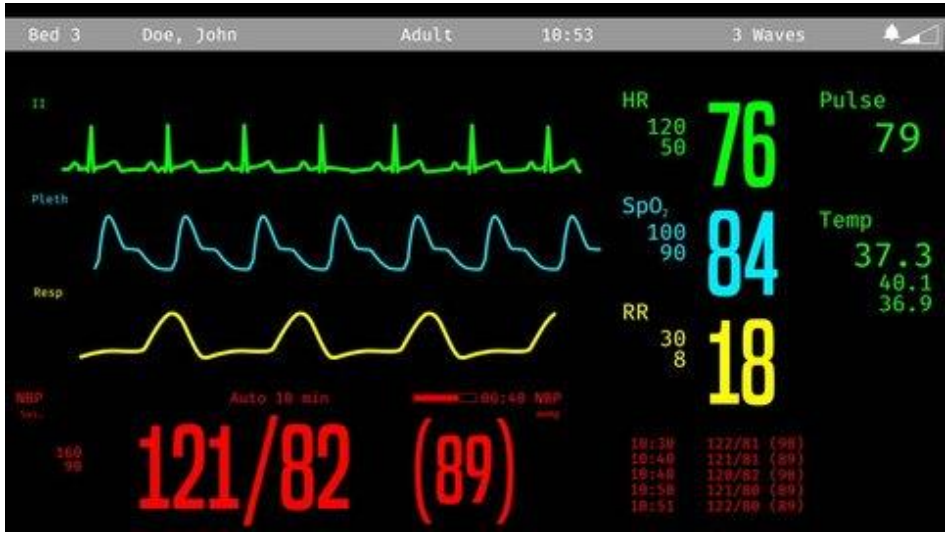
Current market value:
$176.00m

Highest market value:
$220.00m
Dec 17, 2018

Last update: May 30, 2022

```
float market_value;
int age;
float height;
String foot;
```

Data types and variables 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%
```

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

Data types

char	int	float/double
Name	Salary	Height
Place	Age	Weight
Food	Student ID	Distance
Word	Heart rate	PI

Variables

Everything is hardcoded!

Useless!!!

Define variables

Printf example

```
#include <stdio.h>

int main()
{
    printf("1+1=2");
    printf("1+1=%d", 1+1);

    return 0;
}
```

Variables

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

```
type variable; /*declare*/
type variable = value; /*initialize*/
```

```
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; //声明+赋值
printf("num = %d", num);
```

Variables

Declare and initialize a variable **separately**

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

Declare and initialize a variable **jointly**

```
int a = 3, b = 4, c = 100;
float f = 3.14;
double d = -1.2321232;
char c = ‘A’;
```

Variables can be constant and casted

Constant variable （常量）

```
const int x = 3;
int y = 5;
y = 10;
x = 6;
```

Cannot change value of constant!!!

Cast variable （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 - BodyMassIndex
  - lowerCamelCase - bodyMassIndex
  - snake_case – body_mass_index

No need to memorize keywords, IDE will warn you!

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

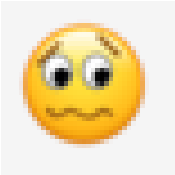
Rules to name variables?

Good names:

int face_num;
int numOfDetectedFaces;
int DetFaceNum;

Prohibited names:

int float;
int main;
int return;



Bad names:

int test1, test2, test3; **// not meaningful**
int jack, marry; **// hard to understand**
int face_num, BodyMassIndex; **// style not consistent**

Variables and data types

Example: create a variable list for student

```
#include <stdio.h>

int main ()
{
        char name[6] = "Tom";
        char gender = 'M';
        int age = 18;
        float height = 1.78;
        int grade = 88;
        return 0;
}
```

Microsoft Visual Studio 调试控制台

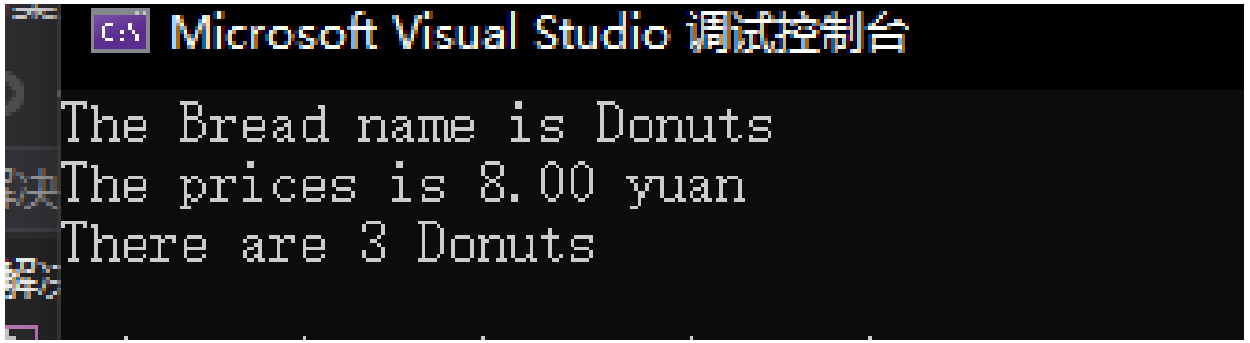
```
name = Tom
gender = M
age = 18
height =1.780000 m
grade = 88
```

int age = 18;
printf("age is %d", age);

Variables and data types

Example: create a variable list for food

```
#include <stdio.h>

int main ()
{
        char name[10] = "Donuts";
        float price = 8.0;
        int num = 3;
        return 0;
}
```



Variables and data types

Example: create a variable list for animal

```
#include <stdio.h>

int main ()
{
        char animal[10] = "Elephant";
        char name[5] = "Elly";
        char gender = 'F';
        int age = 3;
        float weight = 2.03;
        return 0;
}
```

Microsoft Visual Studio 调试控制台

```
The Elephant 's name is Elly
gender = F
age = 3
weight =2.030000 t
```

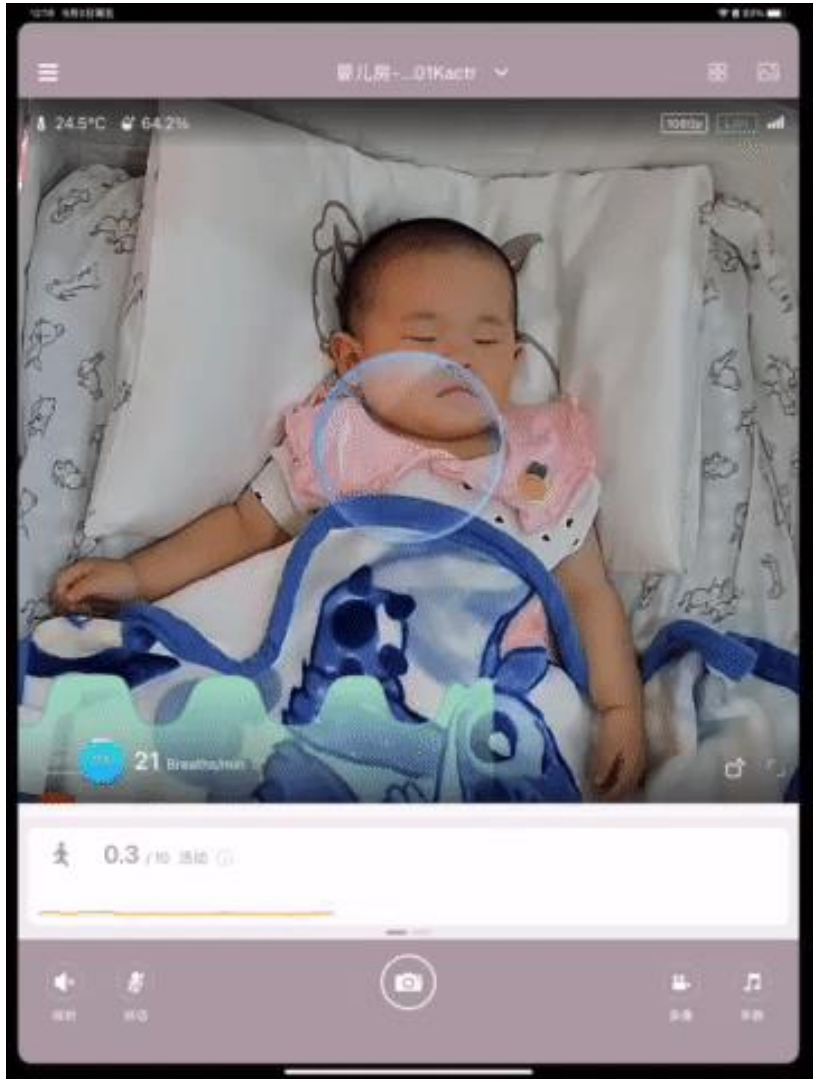
Content

Operations in life

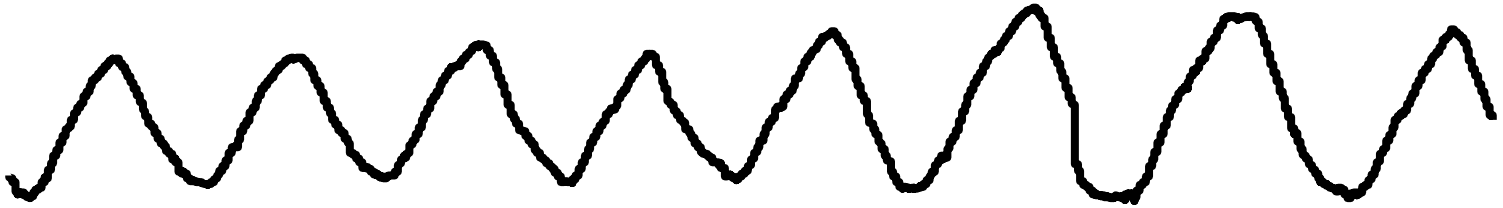
Operations in life


```
capture >> img;//read one frame of the video
if (img.empty())
    break;

mum_of_frame++;
cout << setw(5) << mum_of_frame << endl;
resize(img, img_resized, Size(img.cols / 2, img.rows / 2), 0, 0, INTER_LINEAR);
cvtColor(img_resized, gray, COLOR_BGR2GRAY);
context = get_context(gray, target_pos, sz, window);

context_f = my_dft(context);


if (mum_of_frame > 1)
{
    cv::mulSpectrums(Hstcf, context_f, context_mul, 0);//.mul of Spectrums

    confmap = my_idft(context_mul);

    split(confmap, confmap_planes);

    minMaxLoc(confmap_planes[0], NULL, NULL, NULL, &max_pos);
    target_pos[0] = cvRound(target_pos[0] - sz[0] / 2 + max_pos.x);
    target_pos[1] = cvRound(target_pos[1] - sz[1] / 2 + max_pos.y);

}
```

Operations in life



Breath calculation



```
// define parameters
int limit = 200, H = floor(im.rows / 2), W = im.cols;

int   x_offset = breath_signal.size() < limit ? 0 : breath_signal.size() - limit;
float x_scale = (float)W / limit;

float y_max = *max_element(breath_signal.begin() + x_offset, breath_signal.end());
float y_min = *min_element(breath_signal.begin() + x_offset, breath_signal.end());
float y_offset = 20;
float y_scale = ((float)H - 2 * y_offset) / (y_max - y_min + 0.00001);

cv::Mat panel = cv::Mat(H, W, CV_8UC3, cv::Scalar(255, 255, 255));

for (int idx = x_offset; idx < breath_signal.size() - 1; idx++)
{
    cv::Point2f pt1((idx - x_offset) * x_scale, (y_max - breath_signal[idx]) * y_scale + y_offset);
    cv::Point2f pt2((idx - x_offset + 1) * x_scale, (y_max - breath_signal[idx + 1]) * y_scale + y_offset);
    cv::line(panel, pt1, pt2, cv::Scalar::all(0), 2, 8, 0);
}
```

Operations in life

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	Relational operators	Logical operators	Assignment operators	Misc operators
+,-,*,/,%	>,>=,!=,<	&&, ||, !	=, +=, -=, *=	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
A++

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

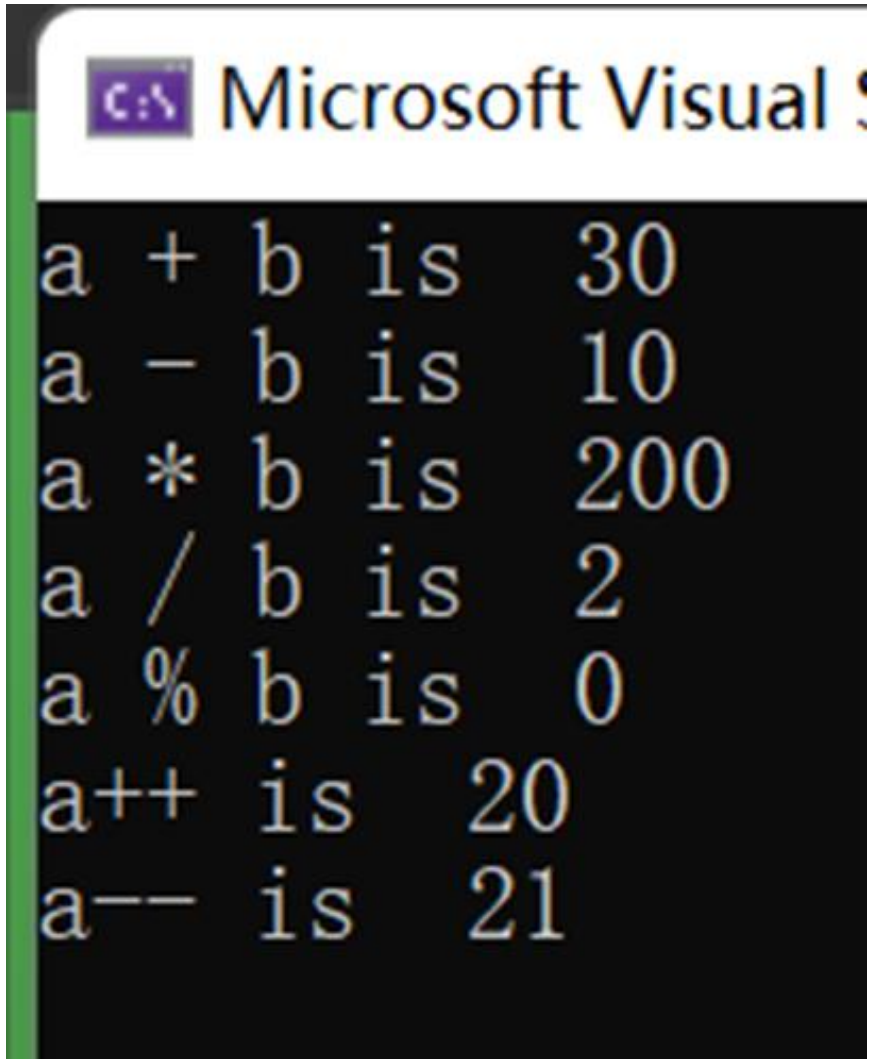
Pre-increment
++A

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

Arithmetic operators

Example 1: basic operations

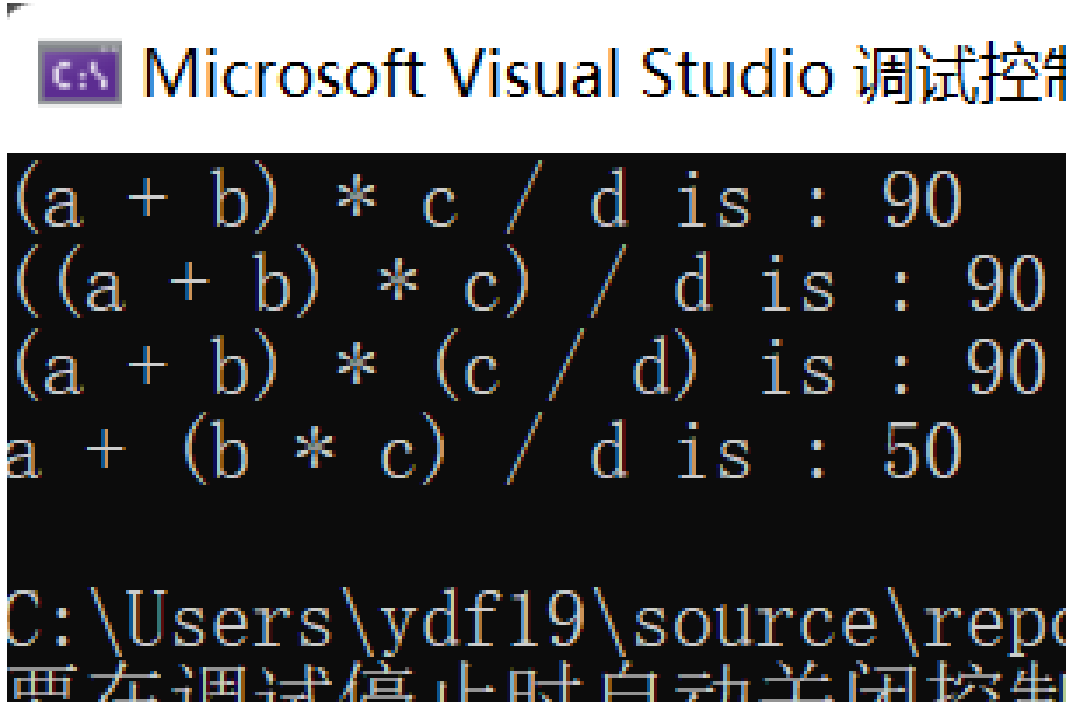
```
#include <stdio.h>
main()
{
    int a = 20, b = 10;
    int c ;
    c = a + b;
    printf("a+b is %d\n", c );
    c = a - b;
    printf("a-b is %d\n", c );
    c = a * b;
    printf("a*b is %d\n", c );
    c = a / b;
    printf("a/b is %d\n", c );
    c = a % b;
    printf("a%%b is %d\n", c );
    c = a++;
    printf("a++ is %d\n", c );
    c = a--;
    printf("a-- is %d\n", c );
}
```



Arithmetic operators

Example 2: operations with ()

```
#include <stdio.h>
main()
{
    int a = 20;
    int b = 10;
    int c = 15;
    int d = 5;
    int e;
    e = (a + b) * c / d;
    printf("(a + b) * c / d is : %d\n", e );
    e = ((a + b) * c) / d;
    printf("((a + b) * c) / d is : %d\n" , e );
    e = (a + b) * (c / d);
    printf("(a + b) * (c / d) is : %d\n", e );
    e = a + (b * c) / d;
    printf("a + (b * c) / d is : %d\n" , e );
}
```



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)

Relational operators

Example 1: comparing integers

```
#include <stdio.h>
main()
{
        int a = 10;
        int b = 20;
        int c = 30;
        int d = 40;
        int e;
        e = a == b;
        printf("10 == 20 ? %d\n",e);
        e = a != b;
        printf("10 != 20 ? %d\n",e);
        e = a > b;
        printf("10 > 20 ? %d\n",e);
        e = a < b;
        printf("10 < 20 ? %d\n",e);
        e = c >= d;
        printf("30 >= 40 ? %d\n",e);
        e = c <= d;
        printf("30 <= 40 ? %d\n",e);
}
```

Microsoft Visual S

```
10 == 20 ? 0
10 != 20 ? 1
10 > 20 ? 0
10 < 20 ? 1
30 >= 40 ? 0
30 <= 40 ? 1

C:\Users\ydf19\s
```

Relational operators

Example 2: comparing floats or characters

```
#include <stdio.h>
main()
{
        float a = 3.14;
        float b = 2.71828;
        char c = 'C';
        char d = 'S';
        int e;
        e = a == b;
        printf("3.14 == 2.71828 ? %d\n",e);
        e = a != b;
        printf("3.14 != 2.71828 ? %d\n",e);
        e = a > b;
        printf("3.14 > 2.71828 ? %d\n",e);
        e = a < b;
        printf("3.14 < 2.71828 ? %d\n",e);
        e = c >= d;
        printf("'C' >= 'S' ? %d\n",e);
        e = c <= d;
        printf("'C' <= 'S' ? %d\n",e);
}
```

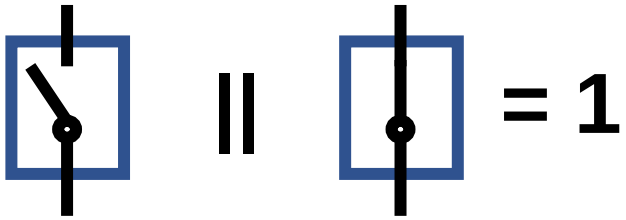
Microsoft Visual Studio 调试控制台

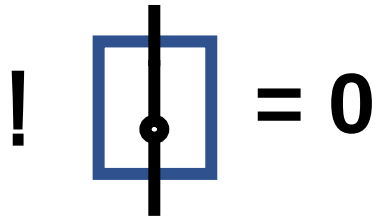
```
3.14 == 2.71828 ? 0
3.14 != 2.71828 ? 1
3.14 > 2.71828 ? 1
3.14 < 2.71828 ? 0
'C' >= 'S' ? 0
'C' <= 'S' ? 1

C:\Users\ydf19\source\repos\He
要在调试停止时自动关闭控制台，
按任意键关闭此窗口. . .
```

Logical operators

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

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

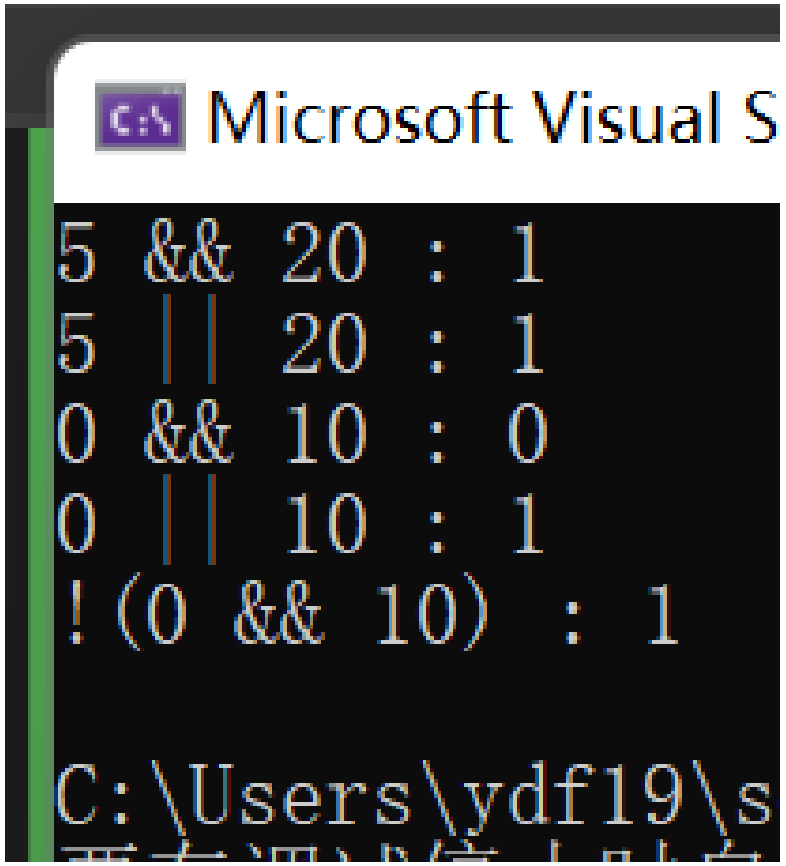




Logical operators

Example 1: comparing integers

```
#include <stdio.h>
main()
{
        int a = 5;
        int b = 20;
        int c;

        c = a && b;
        printf("5 && 20 : %d\n",c);
        c = a || b;
        printf("5 || 20 : %d\n",c);

        a = 0;
        b = 10;
        c = a && b;
        printf("0 && 10 : %d\n", c);
        c = a || b;
        printf("0 || 10 : %d\n", c);
        c = !(a && b);
        printf("!(0 && 10) : %d\n", c);
}
```



Assignment operators

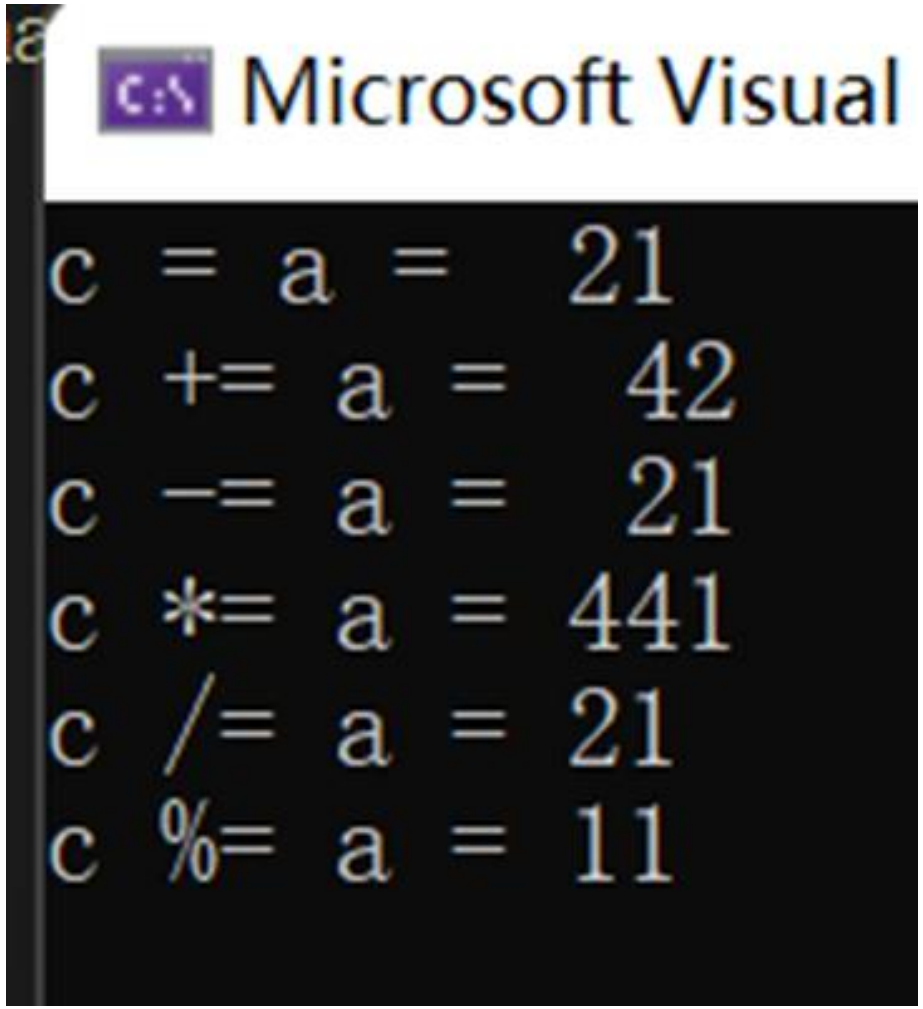
Define two variables: int A = 5, B = 3;

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

Assignment operators

Example 1: assignment of an integer

```
#include <stdio.h>
main()
{
        int a = 21;
        int c;
        c = a;
        printf("c = a = %d\n", c);
        c += a;
        printf("c += a = %d\n", c);
        c -= a;
        printf("c -= a = %d\n", c);
        c *= a;
        printf("c *= a = %d\n", c);
        c /= a;
        printf("c /= a = %d\n", c);
        c = 200;
        c %= a;
        printf("c %%= a = %d\n", c);
}
```



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 = -2072708912 &B = -1602356112
?	Conditional expression	int flag = A>0 ? 1:0;
*****	Pointer points to a variable	*A, *B

Few other important operators supported by C Language.

Miscellaneous operators

Example: use of sizeof(), ?

```
#include <stdio.h>
main()
{
    int a = 10;
    float b = 3.14;
    printf("Storage size for int : %d \n", sizeof(a));
    printf("Storage size for float : %d \n", sizeof(b));
}
```

Microsoft Visual Studio 调试控制台

```
Storage size for int : 4
Storage size for float : 4

C:\Users\ydf19\source\repos
要在调试停止时自动关闭控制台
按任意键关闭此窗口. . .
```

```
#include <stdio.h>
main()
{
        int a = 10, b = 20;
        int c;
        c = a > b ? 1 : 0;
        printf("10 > 20 ? :%d\n", c);
        c = a < b ? 1 : 0;
        printf("10 < 20 ? :%d\n", c);
}
```

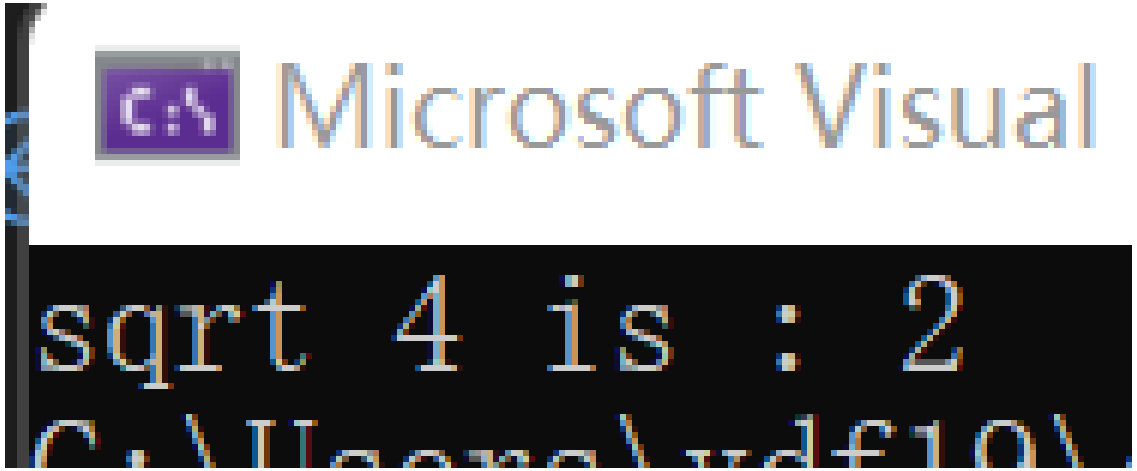
Microsoft Visual Studio 调试控制

```
10 > 20 ? :0
10 < 20 ? :1

C:\Users\ydf19\source\repo
要在调试停止时自动关闭控制
按任意键关闭此窗口. . .
```

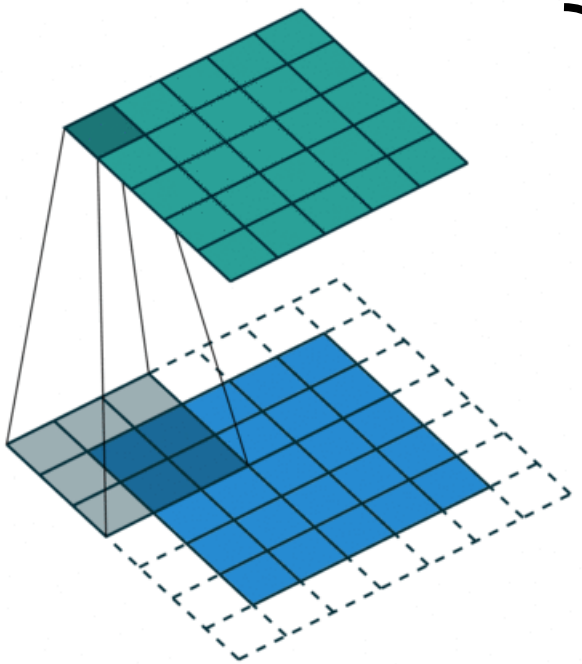
Miscellaneous operators

Example: sqrt()

```
#include <stdio.h>
#include <math.h>

main()
{
      int a = 4;

      int b = sqrt(a);
      printf("sqrt 4 is : %d", b);
}
```



Operators are fundamentals of AI



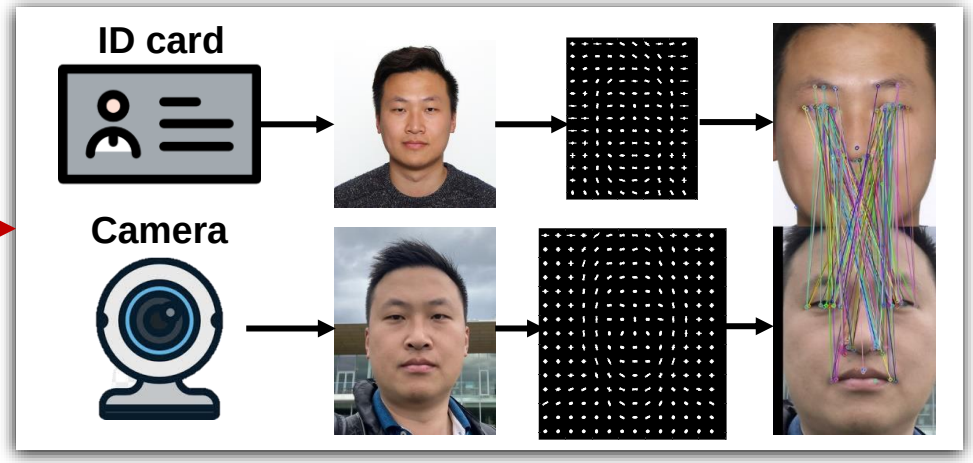
1 ×1	1 ×0	1 ×1	0	0
0 ×0	1 ×1	1 ×0	1	0
0 ×1	0 ×0	1 ×1	1	1
0	0	1	1	0
0	1	1	0	0

Image

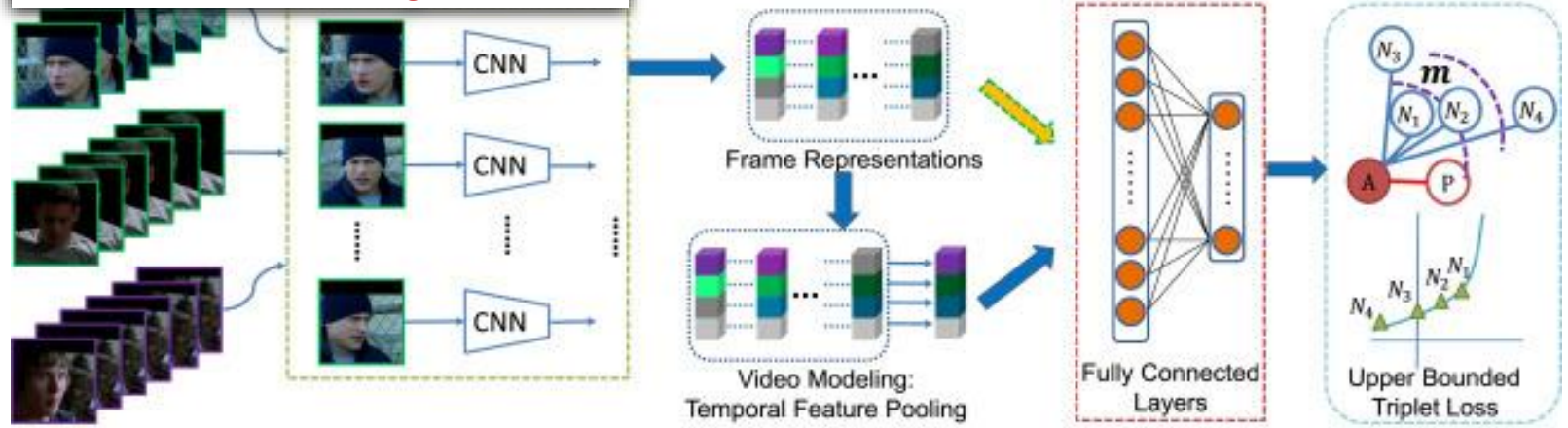
4		

Convolved Feature

Every time when entering “深圳北站”, AI happens



Ideas, maths & algorithms

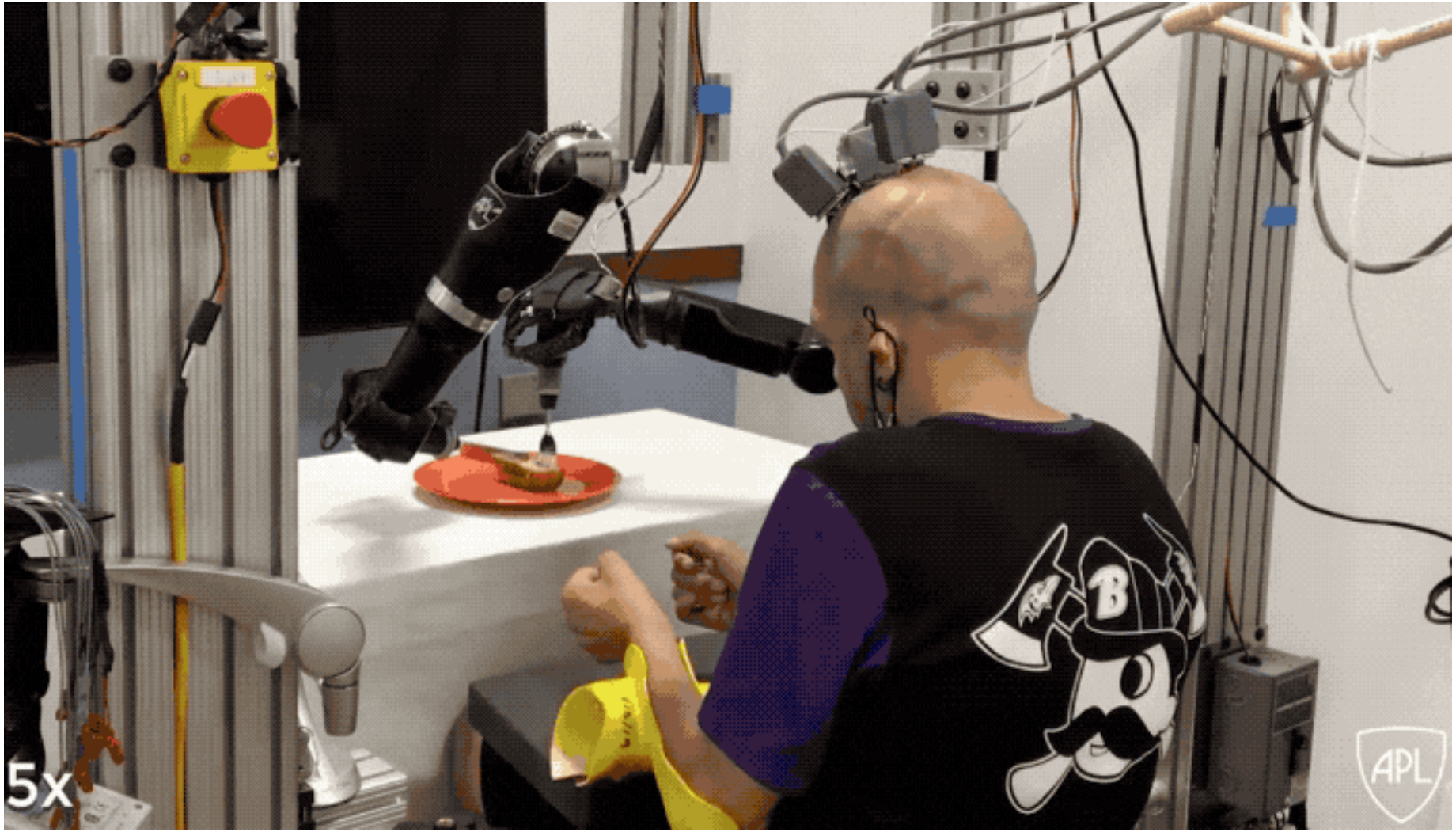


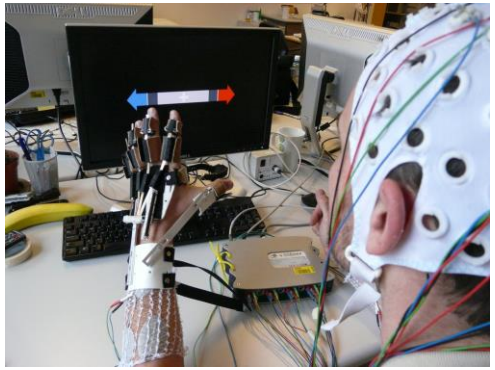
Content

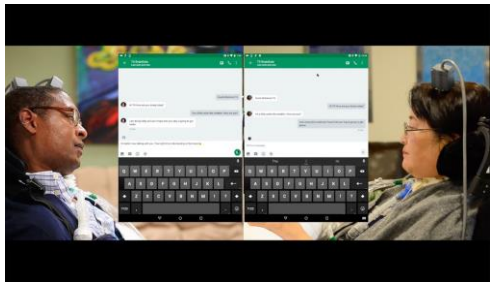
I/O in life

I/O in life

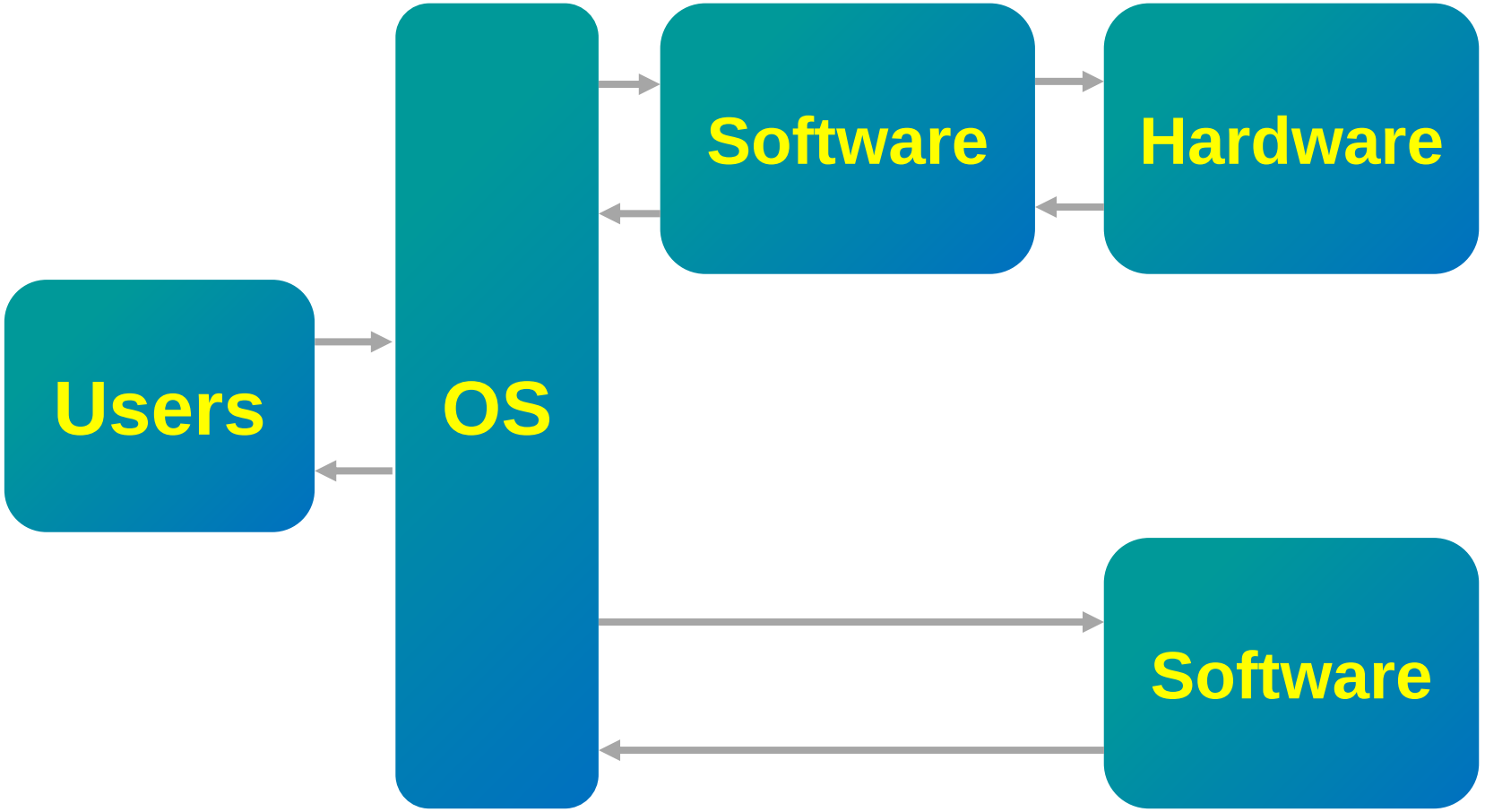
I/O in life





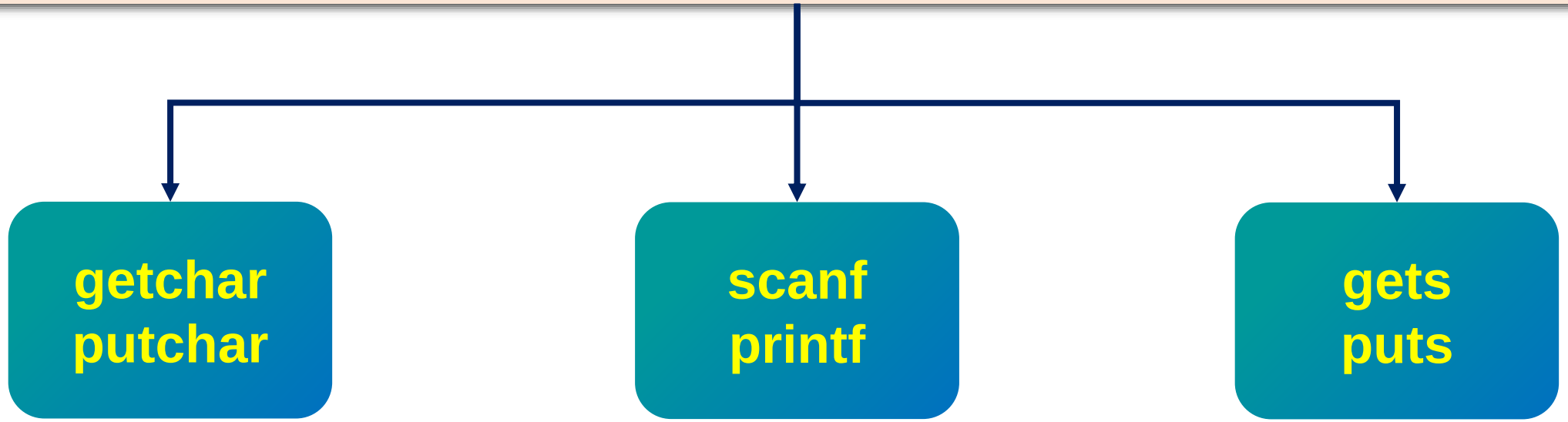


I/O in life



I/O in program

I/O defines how machine reads human's input and put on screen.



getchar() and putchar()

- **getchar()** reads the next available single character and returns an integer representing the character in ASCII table.
- **putchar()** puts the passed character on the screen.

```
int c = getchar();
putchar(c);
```

It reads and puts a single character!!!

getchar() and putchar()

Example 1: input an integer

```
#include <stdio.h>

int main( )
{
  int c;
  printf( "Enter a value :");
  c = getchar( );
  printf( "\nYou entered: ");
  putchar( c );
  return 0;
}
```

```
Enter a value :4

You entered: 4
```

getchar() and putchar()

Example 2: input a character

```
#include <stdio.h>

int main()
{
    char character;

    printf("Enter a character:");
    character = getchar();
    printf("character = ");
    putchar(character);

    return (0);
}
```

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

getchar() and putchar() (demos)

Example 3: input two characters

```
#include "stdio.h"
int main()
{
    char c,d;
    printf("please input two
characters:\n");
    c=getchar();
    putchar(c);
    putchar('\n');
    d=getchar();
    putchar(d);
    putchar('\n');
    printf("character1 = %c\n",c);
    printf("character2 = %c\n",d);
    return(0);
}
```

```
please input two characters:
sd
s
d
character1 = s
character2 = d
```

gets() and puts()

- **gets()** reads a string (a group of characters) from user and puts it into a buffer
- **puts()** shows the string (a group of characters) on the screen

```
gets(char *s);
puts(char *s);
```

It reads and puts a group of characters!!!

gets() and puts()

Example: input a group of characters

```
#include <stdio.h>
int main( )
{
  char str[20];
  printf( "What's your name?\n");
  gets( str );
  printf( "\nYour name: ");
  puts( str );
  return 0;
}
```

```
What's your name?
Alex

Your name: Alex
```

scanf() and printf()

- **scanf()** reads the user input stream and scans it according to the provided format
- **printf()** writes to the output stream according to the format

```
scanf([formatted text], [arguments]);
printf([formatted text], [arguments]);
```

Formatted by specifiers
- **%d int**
- **%f float**
- **%c char**

f means formatted!!!

scanf() and printf()

Example 1: input 2 integers in char and int formats

```
#include <stdio.h>
int main( )
{
char str[100];
int i;
printf( "Enter two value :");
scanf("%s %d", str, &i);
printf( "\nYou entered: %s, %d ", str,
i);
return 0;
}
```

```
Enter two value :67 76

You entered: 67, 76
```

scanf() and printf()

Example 2: input 2 integers and make calculation

```
#include<stdio.h>
int main(void)
{
  int num1;
  int num2;
  int num3=0;
  printf("please enter number1:");
  scanf("%d",&num1);
  printf("please enter number2:");
  scanf("%d",&num2);
  num3=num1+num2;
  printf("number1 + number2 =
%d\n",num3);
  return 0;
}
```

```
please enter number1:4
please enter number2:5
number1 + number2 = 9
```

scanf() and printf()

Example 3: input different types of data

```
#include<stdio.h>
int main(void)
{
  int a;
  char ch;
  float b;

  scanf("%d %c %f",&a,&ch,&b);
  printf("a = %d, b = %.2f,ch =
%c\n", a, b, ch);
  return 0;
}
```

```
3 h 0.9
a = 3,  b = 0.90, ch = h
```

Summary

1. Bit and byte
2. Data types and variables
3. Operations
4. I/O

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**
- Time to write you C program using calculations with I/O

Homework

1. Write following base-10 numbers in base-2 format: 10, 1024 and 1025. Write the following base-2 numbers in base-10 format: 11110 and 1011101.

2. Write a program that prints the minimum, maximum, average and standard deviation of 5 floats.
   a. use “scanf()” to read the number
   b. test input : 23.5 47 -20.1 13 36

3. Write a program that reads the radius of a circle and outputs perimetric and area.
   a. use “scanf()” to read the radius
   b. PI = 3.14
   c. test input : radius of the circle is 3

Homework

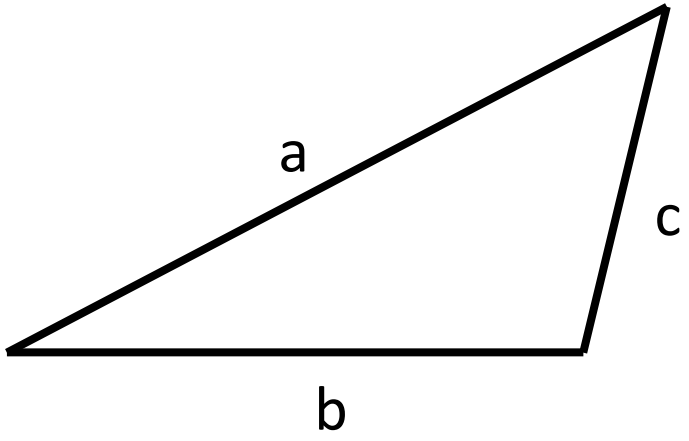
4. Write a program that can find the roots of the equation : $ax^2 + bx + c = 0$.
   a. use “scanf()” to read a,b and c
   b. test input : int a = 4, b = 5, c = 1
   c. If there is no solution, output “-1”

5. Write a program that reads the length of three sides of triangle, outputs its area.
   a. use Heron’s formula to calculate the area
   b. use “?” to check if a triangle can be created
   c. use “scanf()” to read length of the three sides
   d. test input : the length of three sides are 10, 15, 24 and 10,15, 25
   e. If there is no solution, output “-1”

Homework

6. **(bonus)** Chicken and rabbit in the same cage: it is known that the total number of chickens and rabbits is n, and the total number of legs is m. Enter n and m, output the number of chickens and rabbits. If there is no solution, output “-1”.

a. use int type for input and output
b. use “scanf()” to read m and n
c. test input : m = 22 n = 7 and m = 25 n = 9



Heron's formula: $S = \frac{1}{4}\sqrt{(a+b+c)(a+b-c)(a+c-b)(b+c-a)}$

Answers of lecture 1 homework

1. printf(“hello world!”);

2. printf(“12345678\n”);
   printf(“张三\n”);

3. printf(“12345678\n张三\n”);

4. printf(“%d + %d = %d”,1,1,1 + 1);

5. printf(“%f * % f = %f”,3.14,3.14,3.14 * 3.14);