東南大學 SCHOOL OF INTEGRATED CIRCUITS, SEU

集成电路学院

计算机科学基础I

——C++ 基础知识

东南大学 集成电路学院 朱彬武

E-mail: bwzhu@seu.edu.cn

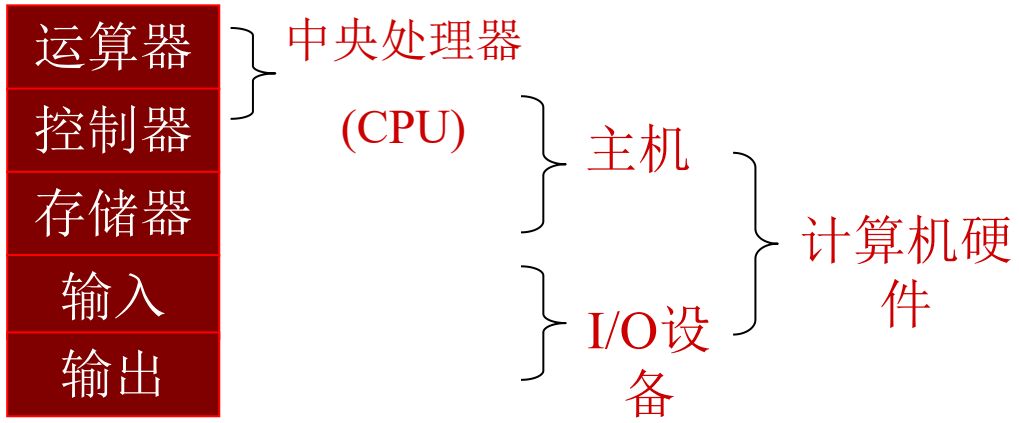
- 计算机的发展历史：计算机是怎么发展到今天的形态
  - 手工
  - 机械
  - 机电
  - 电子
  - 晶体管
  - 集成电路
  - 超大规模集成电路

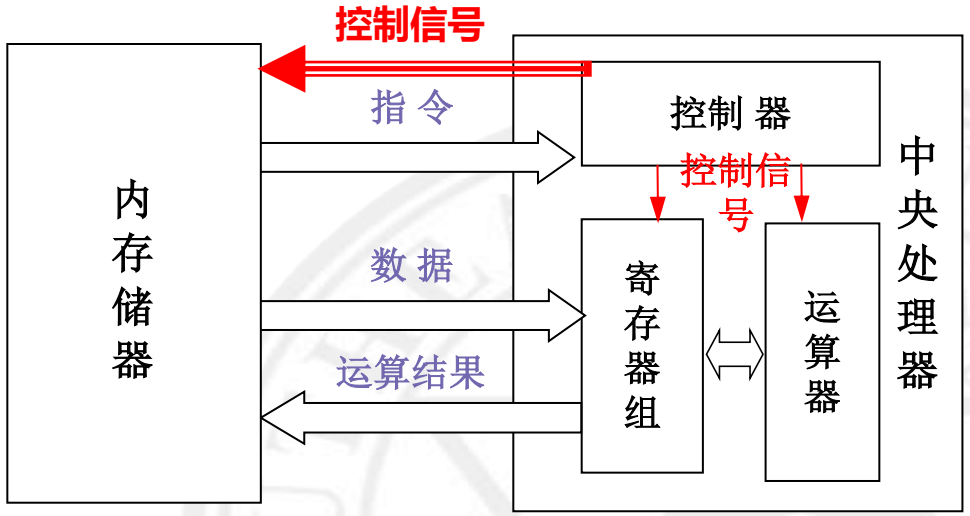
计算速度
越来越快

- 计算机中的信息表示：计算机如何理解人类世界的信息
  - 二进制
    - 进制的基本规律（逢N进1）：

      十进制 10 = 二进制 1010 = 八进制 12 = 十六进制 A
  - 不同进制之间的转换
    - 二进制、八进制、十进制、十六进制
  - 特殊信息的二进制表示：
    - 有符号数（补码）
    - 浮点数（IEEE 754标准，类比科学计数法）
    - 文字（ASCII、Unicode）

- **计算机的通用架构：计算机如何执行人类的指令，实现通用性**
  - **冯诺依曼架构：将程序存储在内存里**





- 存储器的金字塔结构
- 常见存储单位的转换

- 更早期专用计算机：功能固定
- 早期通用计算机：重新插线修改运算器电路结构
- 现代通用计算机：通过控制信号启用运算器相应电路
  - 控制信号：加法、减法、增量+1、减量-1.....

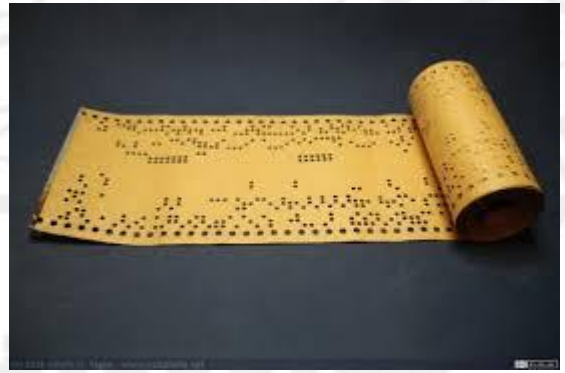
CHAPTER 1

C++基础知识

程序员如何和计算机交流?

编程的早期形态

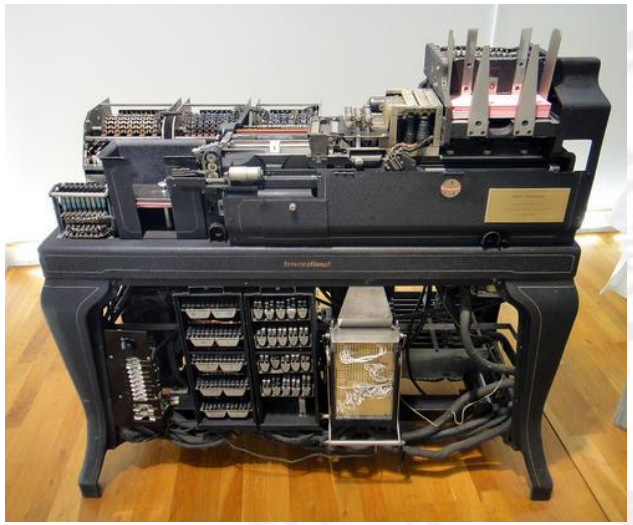
- 雅卡尔织布机：简化了复杂图案的纺织品的生产过程
  - 使用织布机前设计需要编织的图案，并根据设计以及相应规则在打孔卡上打孔
  - 根据孔的有无来控制经线与纬线的上下关系



这种使用可更换的打孔卡来编织纺织品花样的原理被认为是计算机硬件历史上的重要一步。

编程语言发展历史

- 诞生：机器语言
  - 二进制指令
  - 早期以接线、穿孔卡片或穿孔纸带形式存储
  - 完全依赖于硬件
- 应用：1955年美国空军的SAGE防空系统，62500张纸带



1935年
IBM穿孔卡片机（美）

完成1+2的计算：

```
0: 55
1: 48 89 e5

4: c7 45 fc 01 00 00 00
b: c7 45 f8 02 00 00 00

12: 8b 45 f8
15: 01 45 fc

18: 5d
19: c3
```

编程语言发展历史

- 汇编语言（Assembly Language）
  - 助记符号
  - 与硬件关系密切
  - 需要汇编器转成机器语言

完成1+2的计算：

```
push rbp
mov rbp, rsp

mov DWORD PTR [rbp-0x4], 0x1
mov DWORD PTR [rbp-0x8], 0x2

mov eax, DWORD PTR [rbp-0x8]
Add DWORD PTR [rbp-0x4], eax

pop rbp
ret
```

编程语言发展历史

- 二进制指令基本格式
  - 操作码
  - 操作数（寻址）
    - 立即寻址
    - 寄存器寻址
    - 存储器寻址

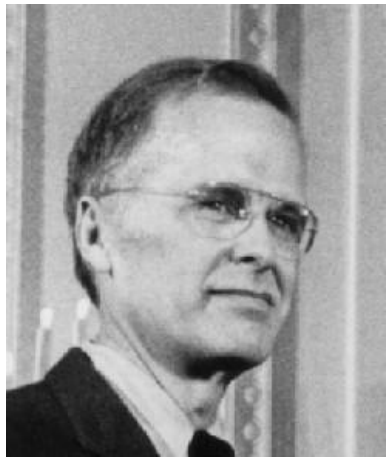
指令格式

MOV	AL, 7	1011 0000 0000 0111
ADD	AL, 10	0000 0111 0000 1010
HLT		1111 0100
(halt)		

编程语言发展历史

- 高级语言
  - 符号语法更利于阅读和设计，更加规范
  - 需要强大的语言转换工具

葛丽丝·霍普，A-0系统的发明者，高级编程语言最早的提出者



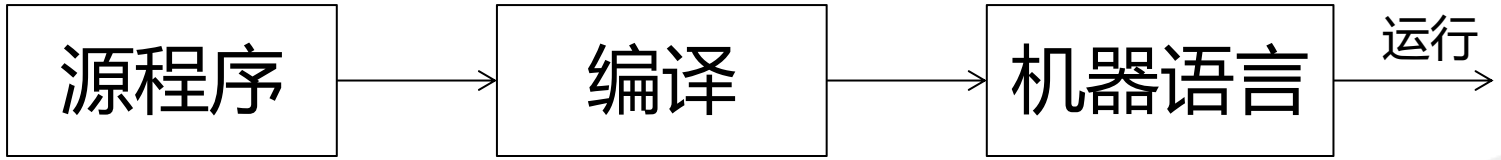
约翰·巴克斯（美国）第一个高级语言FORTRAN的设计者

完成1+2的计算：

```
int main()
{
    int a = 1;
    int b = 2;
    a = a + b;
}
```

编程语言类型——执行方式

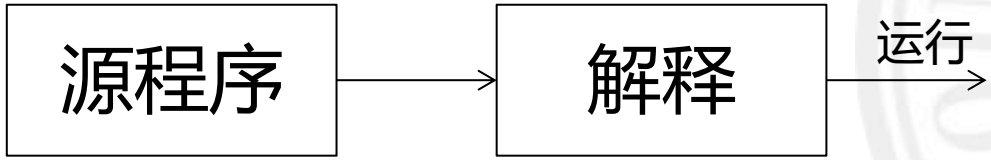
- **编译模式（Compile）**
  - 借助一个程序，将源程序一次性转换为计算机能懂的语言（机器语言），然后这个机器语言写的程序就可以直接运行。



运行程序直接执行机器码

类比：提前写好翻译稿

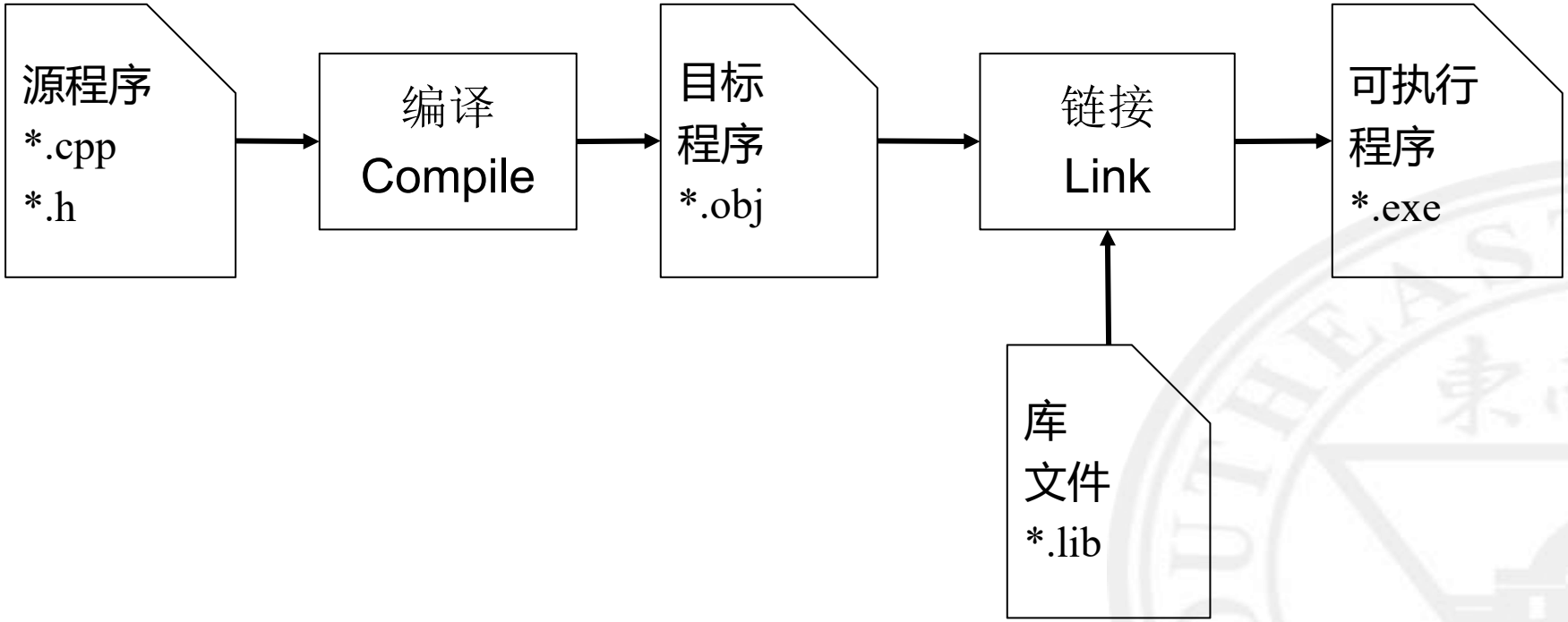
- **解释模式（Translate）**
  - 借助一个程序理解你写的程序，然后按照你的要求执行



运行程序要逐条解释

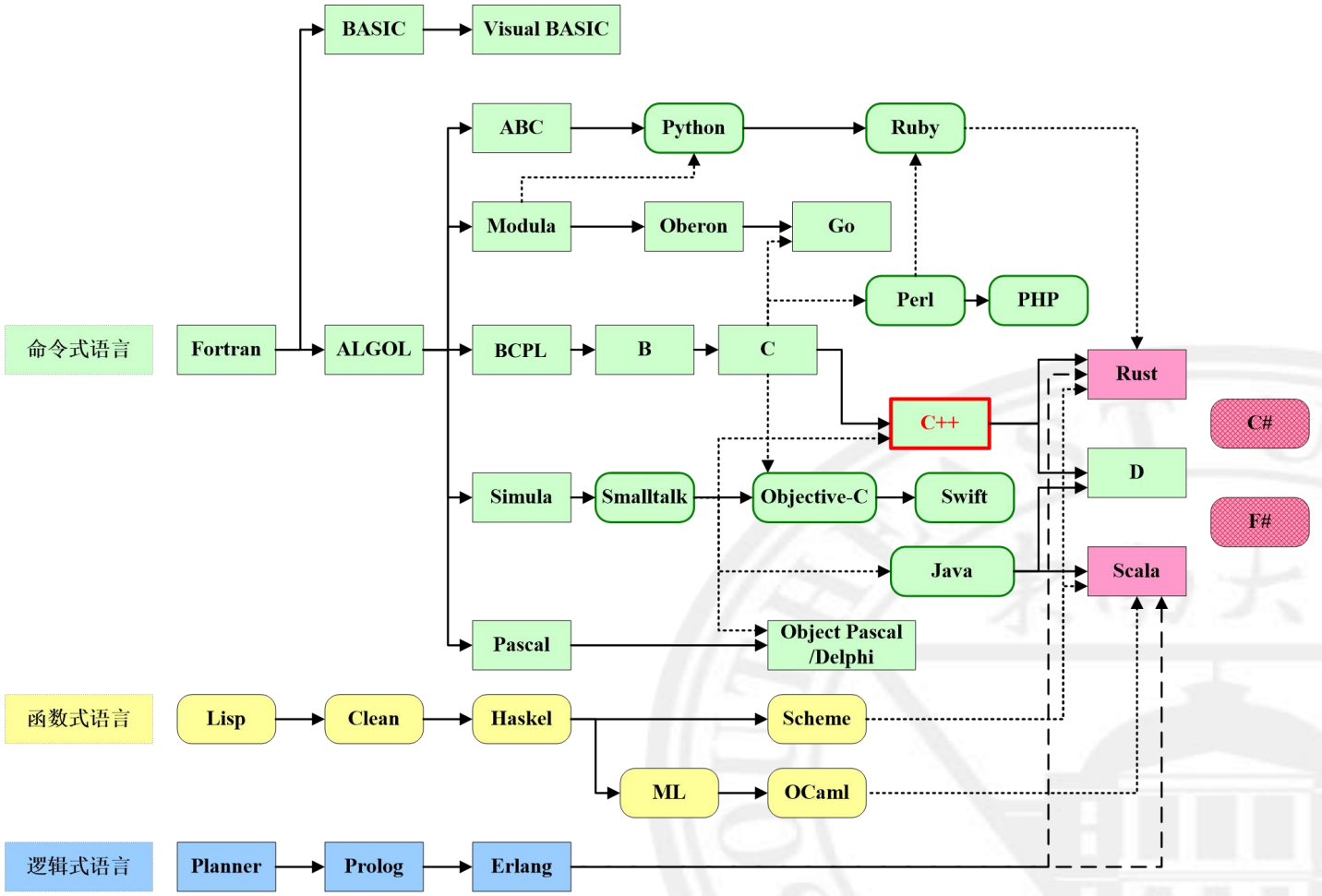
类比：现场口译

C++执行方式：编译执行



编程语言发展历史

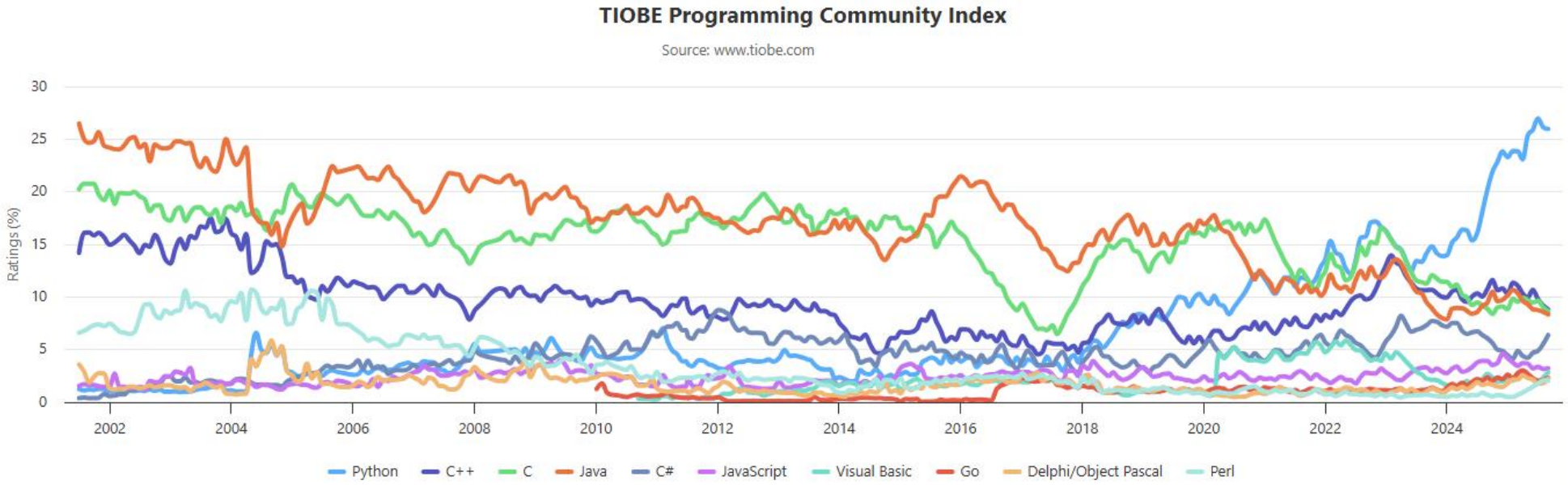
1957 FORTRAN
1960 COBOL
1960 LISP
1965 BASIC
1971 PASCAL
1973 C
（Ken Thompson，Dennis Ritchie）
1975 SmallTalk
1983 C++
（Bjarne Stroustrup）
1986 Objective-C
1990 Python
1995 Java
2000 C#
2009 Go
2012 Rust
2014 Swift
……



C++语言的诞生之路

- 在 1960 年代，AT&T 贝尔实验室开发了一种操作系统 Multics，但项目庞大复杂，最终被放弃。
- 1969 年，Ken Thompson 在 PDP-7 上开发了一个简化的操作系统 UNIX（UNICS改名），用 汇编语言编写，效率高但不够灵活。
- 为了提高可移植性和开发效率，Thompson 在 BCPL（Basic Combined Programming Language） 的基础上，设计了一种简化语言 B（1969 年）。
- Dennis Ritchie 在 B 语言的基础上扩展，增加了数据类型和结构化编程特性，设计出了 C 语言（1972 年）。最初的 C 语言就是为 UNIX 操作系统编写而生的。
- Bjarne Stroustrup（本贾尼·斯特劳斯特卢普）在研究计算机模拟（尤其是分布式系统仿真）时，发现 C 语言的效率很高，但在抽象表达方面不足；而 Simula 语言（第一门面向对象语言）虽然有很好的抽象机制，但运行效率太低。于是他结合二者的优点，开始在 C 的基础上扩展，引入“类”的概念，最初被称为 “C with Classes”。
- 1983 年：Stroustrup 将其语言更名为 C++，名字来源于 C 的自增运算符 ++，寓意“在 C 的基础上改进和增强”。

编程语言的使用变化



Python：人工智能领域的主流，学习成本低

C/C++：工业、底层软件的主流，虽未登顶
但经久不衰，效率至上，学习成本高

来源：https://www.tiobe.com/tiobe-index/

现代编程语言的主要特点

- 现代编程语言在语法上的差异很小（C-like语言）

◆ C++

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

int main() {
    cout << "Hello, World!" << endl;
    return 0;
}
```

◆ Python

```python
print("Hello, World!")
```

◆ Rust

```rust
fn main() {
    println!("Hello, World!");
}
```

- 语言的能力/适用领域主要是由库和传统决定

C++语言概述

C++发展历史

- **发展历史**
  - 1983年诞生，改进C语言，增加了类
  - 1994年ANSI标准化
  - 1998年ISO标准化（ISO14882），C++98
  - 2003年C++03
  - 每隔3年修订一版，C++11/C++14/C++17/C++20
  - 最新标准，C++23

Bjarne Stroustrup（丹麦）
C++之父

C++应用领域

- 操作系统（Windows、MacOS、Linux）
- 游戏开发（OpenGL、Unreal Engine）
- 嵌入式系统（智能手表、路由器、无人机）
- 工业软件（AutoCAD、ANSYS、Matlab、Altium Designer）
- 人工智能（PyTorch、TensorFlow、OpenCV）
- 桌面软件（Office、Chrome、Photoshop）

C++开发工具

- C++需要被编译才能运行，所以我们需要
  - 编辑器
  - 编译器

- 或者IDE（Integrated Development Environment，集成开发环境），适合新手

C++开发工具

- Microsoft Visual C++ 2010学习版
  - 对标全国计算机等级考试
  - 期末上机考试使用
  - 安装方式和基本使用文档已在课程群上传

IDE其他选择

- Visual Studio社区版（免费，体积大，只有Windows版本）

https://visualstudio.microsoft.com/zh-hans/downloads/

- CLion（需要提交学信网的学籍认证报告，支持跨平台）

Sorry, we can't accept your email address because we are not able to identify if the people from this email domain are truly students or not.

This could have happened for the following reasons:

— Your school allows registering an email address there even if you are not a student of the college.

— Someone from the school sold or shared their account details or activation code publicly, which has led to a broad account misuse and license terms violation.

If you are a student of this college, please apply with an official document.

编辑器+编译器

- 常用编辑器（支持多平台）
  - VS Code（轻量，插件丰富，推荐）
  - Sublime
  - Vim（学习成本高，命令行文本编辑器）
- 编译器
  - MinGW（Windows）
  - GCC（Linux，MacOS（实际底层是LLVM））
- 可以通过配置编辑器（如VS Code）实现类似IDE的效果

C++——输出、输入、变量

第一个C++程序


```
#include <iostream>

int main(){
        std::cout <<  “Hello World”  <<std::endl;
        return 0;
}
```

程序框架


```
#include<iostream>

int main(){
    
    return 0;
}
```

- 本课程中所有的程序都包含这一段基本框架
- 直到学到函数之前，我们都是在这个框架内部完成代码编写

命名空间


```
std::cout << "Hello World" << std::endl;
```

- std是C++的标准命名空间（namespace），cout和endl都在std的命名空间内
- 也可以写作下面这个形式，告诉编译器使用std里的名字，这样就不需要每次都加上std前缀

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

int main(){
    cout << "Hello World" << endl;
}
```

- 命名空间的作用是防止名字冲突，在小程序或者学习阶段这么写没问题，但在大型项目里最好不要全局使用，因为可能会引起名字冲突。

C++做输出


```
std::cout << "Hello World" << std::endl;
```

- cout作用可以看作是把后面的字符串输出到标准输出设备（显示器），cout 要结合 << 使用，<<是插入运算符
- endl的作用是换行

当程序出现错误


```
#include <iostream>


int main(){
        std::cout <<  “Hello World!” <<std::endl
        return 0;
}
```

- 学会看编译器给出的报错信息

最容易犯的错误

不要用中文，尤其是标点符号，包括逗号，分号，括号，感叹号等等，极其容易犯的错误！！

C++常用数学运算符号

常用数学运算	C++符号	含义
+	+	加
-	-	减
×	*	乘
÷	/	除
mod	%	取余
()	()	括号

运算符分类

- 按运算性质分类
  - 算术运算符：+、-、*、/、%
  - 关系运算符：>、<、==、!=、<=、>=
  - 位运算符：&、|、^、~、<<、>>
  - 逻辑运算符：&&、||、！
  - 赋值运算符：=、+=、-=、*=、/=、%=、>>=、<<=
  - 其他
    - 顺序运算符（,）、括号运算符()、自增/自减运算符（++、--）
    - 下标运算符[ ]、地址运算符（*、&）
    - 成员运算符（.、->）、域运算符（::）
    - 动态变量运算符（new、delete）
    - 类型运算符（sizeof、typeid）等

计算输出


```
std::cout << 23489 + 12345 << std::endl;
std::cout << 23489 - 12345 << std::endl;
std::cout << 23489 * 12345 << std::endl;
std::cout << 23489 / 12345 << std::endl;
std::cout << 23489 % 12345 << std::endl;
```

为什么是1？

有点麻烦：

- **需要多次输入两个很长的数**
- **换两个其他的数字计算的时候需要手动修改每一个地方**

变量引入


```
int a = 23489;
int b = 12345;
std::cout << a + b << std::endl;
std::cout << a - b << std::endl;
std::cout << a * b << std::endl;
std::cout << a / b << std::endl;
std::cout << a % b << std::endl;
```

- **定义一次，多次使用**
- **换两个其他的数字计算的时候只需要修改a和b的值**

变量


```
int a = 23489;
int b = 12345;
```

- int a = 23489; 定义了一个变量，变量的名字是a，类型是int（整数，integer），初始值是23489；
- 变量是一个保存数据的地方，当我们需要在程序里保存数据时，就需要变量来保存它。用一个变量保存了数据，它才能参加到后面的计算中。

变量名字


```
int a = 23489;
```

- int a = 23489定义了一个名字叫做a的变量
- 变量需要一个名字，变量的名字是一种“标识符”，意思是它是用来表示不同的东西
- 变量命名的基本原则
  - 由英文字母（区分大小写：a~z，A~Z）、数字字符（0~9）、下划线（_）和通用字符集字符组成
  - 数字字符不能作为首字符
  - 关键字（保留字）不能用作标识符
  - 唯一性
- 好的命名直接影响代码的可读性、可维护性和协作效率。

命名风格（参考）

- 一般变量：小写开头的驼峰命名：studentNumber / 小写字母+下划线 student_number
- 类：大写开头的驼峰命名：StudentInfo
- 方法/函数：小写开头的动词，getStudentNumber
- 常量：全大写加下划线分隔：MAX_VALUE

供参考，不同人不同企业都有不同的命名风格。

C++关键字

C++ keywords

This is a list of reserved keywords in C++. Since they are used by the language, these keywords are not available for re-definition or overloading. As an exception, they are not considered reserved in attributes (excluding attribute argument lists).(since C++11)

A - C	D - P	R - Z
alignas (C++11)	decltype (C++11) (2)	reflexpr (reflection TS)
alignof (C++11)	default (1)	register (3)
and	delete (1)	reinterpret_cast
and_eq	do	requires (C++20)
asm	double	return
atomic_cancel (TM TS)	dynamic_cast	short
atomic_commit (TM TS)	else	signed
atomic_noexcept (TM TS)	enum (1)	sizeof (1)
auto (1) (3) (4) (5)	explicit	static
bitand	export (1) (4)	static_assert (C++11)
bitor	extern (1)	static_cast
bool	false	struct (1)
break	float	switch
case	for (1)	synchronized (TM TS)
catch	friend	template
char	goto	this (5)
char8_t (C++20)	if (3) (5)	thread_local (C++11)
char16_t (C++11)	inline (1) (3)	throw (3) (4)
char32_t (C++11)	int (1)	true
class (1)	long	try
compl	mutable (1)	typedef
concept (C++20)	namespace	typeid
const	new	typename (3) (4)
consteval (C++20) (5)	noexcept (C++11)	union
constexpr (C++11) (3)	not	unsigned
constinit (C++20)	not_eq	using (1) (4)
const_cast	nullptr (C++11)	virtual
continue	operator (1)	void
contract_assert (C++26)	or	volatile
co_await (C++20)	or_eq	wchar_t
co_return (C++20)	private (4)	while
co_yield (C++20)	protected	xor
	public	xor_eq

https://en.cppreference.com/w/cpp/keywords.html

- 不需要记忆，随着学习的深入会慢慢了解每一个关键字
- 即使不小心把关键字用作变量的名字，编译器也会提醒你

变量定义

- 变量定义的一般形式就是：
  - <类型名称> <变量名称>;
- int income;
- int tax;
- int income, tax;

变量初始化


```
int a = 23489;
```

- int a = 23489定义了一个初始值为23489的变量
- 变量初始化是在创建变量的同时给它赋予一个初始值。
- 最常用的初始化方式

<类型名称> <变量名称> = <初始值>

- 组合变量定义的时候，也可以在这个定义中单独给单个变量赋初值

```
int a = 23489, b = 12345;
```

C++中的“=”

<类型名称> <变量名称> = <变量初始值>

- 和数学不同，a=b在数学中表示关系，即a和b的值一样；而在程序设计中，a=b表示要求计算机做一个动作：将b的值赋给a。
- 关系是静态的，而动作时动态的。在数学中，a=b和b=a是等价的，而在程序设计中，两者的意思完全相反。

```
int x = 2;
int y = x;
```

如果没有初始化呢?


```
int a = 0;
int b;
b = a + 10;
std::cout << b << std::endl;
```

```
int a;
int b;
b = a + 10;
std::cout << b << std::endl;
```

- 冯诺依曼结构告诉我们，变量存储在内存里，当你定义一个变量时，编译器会在内存中给它分配一块空间；
- 这块内存可能之前被程序或系统使用过，所以如果没有初始化，a的值就是内存里原本的残留数据；
- **变量一般需要先赋予初值再使用。**

赋值和表达式

- 赋值并不只在初始化中才会用到，当我们在定义完变量以后要修改变量的值，往往就会用到“=”

```
int a = 23489, b = 12345;
int temp = 0;
temp = a;
a = b;
b = temp;
```

- “=”是赋值运算符，有运算符的式子就叫做**表达式**（expression）

变量类型


```
int a = 23489;
```

为什么需要类型？回想上节课讲的二进制。（内存中存的只有0和1）

- int a = 23489定义了一个类型为int的变量；
- C++是一种**有类型的语言**，所有的变量在使用之前必须定义或声明，所有的变量必须具有确定的数据类型。
- 数据类型在变量中可以存放什么样的数据，变量中只能存放指定类型的数据
- 程序运行过程中也不能改变变量的类型。

变量类型


```
int a = 10;
cout << "a = " << a << endl;
a = 3.14;
cout << "a = " << a << endl;
```

- 当你尝试给a赋一个非整数的数3.14时，编译器会强制将3.14变成整数
- a在整个程序中都是int类型

写死的程序


```
int a = 23489;
int b = 12345;
std::cout << a + b << std::endl;
std::cout << a - b << std::endl;
std::cout << a * b << std::endl;
std::cout << a / b << std::endl;
std::cout << a % b << std::endl;
```

不灵活，每次修改数据之后都要重新编译

更灵活的程序——支持输入


```
int a, b;
cout << "请输入你要做四则运算的两个数字，用空格分隔" << endl;
cin >> a >> b;
cout << "计算结果如下:" << endl;
cout << a << " + " << b << " = " << a + b << endl;
cout << a << " - " << b << " = " << a - b << endl;
cout << a << " * " << b << " = " << a * b << endl;
cout << a << " / " << b << " = " << a / b << endl;
```

C++做输入


```
cout << "请输入你要做四则运算的两个数字，用空格分隔" << endl;
cin >> a >> b;
```

- 输入也在终端窗口中，当你输入完之后，需要按下Enter键来告诉程序输入结束，可以开始读取
- cin是标准输入流对象，用来读取用户输入；>>称为提取运算符，将输入的值存入变量，遇到空白、Tab或回车，提取操作会停止
- 一般搭配cout使用，使用cout输出提示信息，引导用户输入

浮点数引入

- **美国人和中国人使用的单位体系不一样，美国人用英制计量单位，中国人用公制计量单位，如果遇到一个美国人告诉你他的包裹重是5磅8盎司，换算成千克是多少?**

- **$(8 + 5 / 16) \times 0.453592 = 3.77$千克**

0.45359237 kg

1磅（lb）精确等于0.45359237千克（kg），该换算基于国际标准定义，自1958年起被美国及英联邦国家广泛采用。 1 2

用C++实现单位转换


```
int pound, ounce;
cout << "请分别输入包裹的英镑和盎司，如输入\"8 5\"表示8英镑5盎司:" << endl;
cin >> pound >> ounce;

cout << "质量是" << (pound + ounce / 16) * 0.453592 << "千克"  << endl;
```

用C++实现单位转换


```
int pound, ounce;
cout << "请分别输入包裹的英镑和盎司，如输入\"8 5\"表示8英镑5盎司:" << endl;
cin >> pound >> ounce;

cout << "质量是" << (pound + ounce / 16) * 0.453592 << "千克"  << endl;
```

问题在哪里?

- 因为两个整数的运算的结果只能是整数，直接截取掉小数的部分（不是四舍五入）
  - 23489/12345=?　10/3*3 = ?
- 10和10.0在C++中是完全不同的数，10.0/3*3=?
- 10.0被称为浮点数

浮点数

- 带小数点的数值。浮点这个词的意思是指小数点是浮动的，是计算机内部表达非整数（包括分数和无理数）的一种方式
- 另一种方式叫做定点数，但是在C++语言中不会遇到定点数
- 人们借用浮点数这个词来表达所有带小数点的数

重新用C++实现单位转换（1）


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

int main(){
    int pound, ounce;
    cout << "请分别输入包裹的英镑和盎司，如输入\"8 5\"表示8英镑5盎司:" << endl;
    cin >> pound >> ounce;

    cout << "质量是" << (pound + ounce / 16.0) * 0.453592 << "千克"  << endl;
}
```

- 当浮点数和整数放到一起运算时，会将整数转换成浮点数，然后进行浮点数的运算

重新用C++实现单位转换（2）


```
double pound, ounce;
cout << "请分别输入包裹的英镑和盎司，如输入\"8 5\"表示8英镑5盎司:" << endl;
cin >> pound >> ounce;

cout << "质量是" << (pound + ounce / 16) * 0.453592 << "千克"  << endl;
```

不同的数据类型不影响变量定义的方式

浮点数类型——double, float

- 除了把16改成16.0，我们也可以把pound和ounce设置为浮点数类型
- C++语言里有两种常用的数据类型用来表示浮点数——double和float
- double的意思是“双”，是“双精度浮点数”的简称，除了double，还有float，float本意为浮点，表示“单精度浮点数”，两者表示的精度不同

$(1001.0111010)_2 = +(1.001011101)_2 \times 2^{(1010)_{2[移]}}$

$(-0.0001010011)_2 = -(1.010011000)_2 \times 2^{(0011)_{2[移]}}$

	单精度	双精度
数符	1	1
阶	8	11
尾数	23	52

思考题

看起来在C++中，整数是浮点数的子集，那我们还需要int类型吗，是否可以把所有的数字都定义为double类型？int类型有没有什么好处？

作业&提醒

- 作业今天会在课程群上传，请及时关注
- 下周有上机课，10.17第6，7节在中心楼上实验课，第8，9节在东南院204上理论课
- 每位同学尽快在自己电脑上熟悉visual c++使用