Computer Fundamentals and Programming

International School of Beijing University of Posts and Telecommunications

Wang Ying
wangy@bupt.edu.cn

北京郵電大學

新的起点，新的征程，我们整装待发

相信，你也有很多的疑惑...

惑？

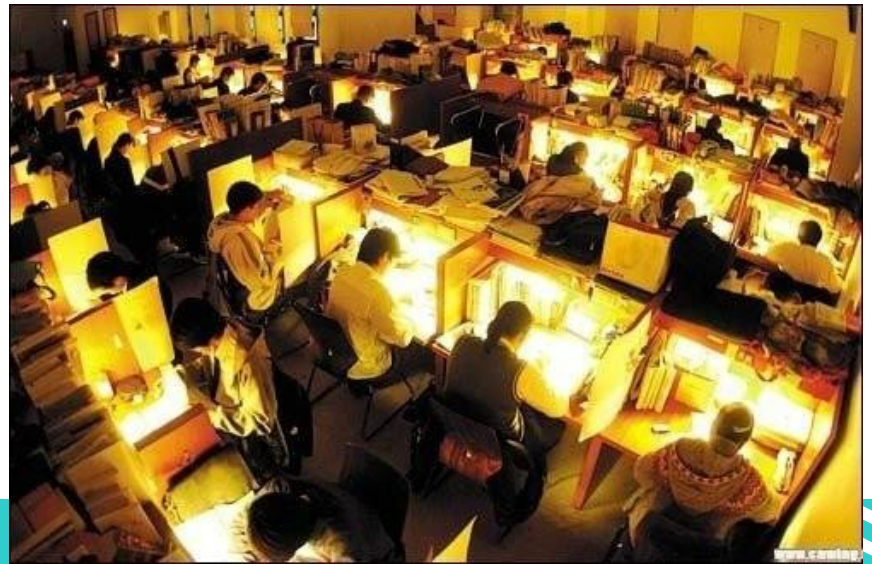
- 大学学习和初高中学习有什么区别?
- 未来四年，我们会学到什么?
- 如何正确度过大学四年?
  - 四年后，读研、出国留学、工作...
- 四年后，我应该具备哪些专业素质?
- 《计算导论与程序设计》这门课讲什么？在整个课程体系中的地位如何？课程重点是什么？如何学好这门课？需要注意哪些问题？
- ……

习总书记强调，青年时代是学习的黄金时期，应该把学习作为首要任务，作为一种政治责任、一种精神追求、一种生活方式，树立梦想从学习开始、事业靠本领成就的观念，让勤奋学习成为青春远航的动力，让增长本领成为青春搏击的能量。

学习不是人生的全部，但连学习都征服不了，你还能做什么？

所有人的成功都不是偶然的，有本领才有机会成功。

习总书记指出，学习贵在勤奋、贵在钻研、贵在有恒；既要惜时如金、孜孜不倦，下一番心无旁骛、静谧自怡的功夫，又要突出主干、择其精要，努力做到又博又专、愈博愈专。



应该如何学？

- 如何正确对待基础知识和热点知识

  作为计算机专业的学生，第一二年基础课程的学习非常重要，要能静下心来听课、看书、练习、思考、体会；不要将精力过多放到计算机操作以及软硬件开发技术上，造成本末倒置。

- 课程学习要成体系

  每学习一门新课程，要设法弄清楚本课程在整个课程体系中的位置，本课程的教学目的和教学重点。每学完一门课程，能将其与之前学习的课程进行联系，做到知识点的融会贯通。

- “师傅领进门，修行在个人”

  高中：老师反复讲解知识；

  大学：自我总结、实践、自学、独立思考、答疑、网络、图书馆、和同学间讨论（*独学而无友，则孤陋而寡闻*）。

关于本课程

课程概述

- 课程名称：计算导论与程序设计
- 面向学院：计算机学院、网络空间安全、国际学院（物联网工程、智能科学与技术专业）
- 授课年级：本科一年级
- 学分及学时：4/64
- 教材及讲义：自编教材+参考书
- 支撑毕业要求指标：1.3、2.1、2.2
  - 计算机学科通识内容，计算求解的基本能力
  - 领域复杂工程问题识别和需求描述
  - 领域复杂工程问题需求，工程学原理级方法分析和抽象

北京郵電大學

课程概述–课程目标

计算为核心，编程为抓手，“理论、抽象、设计”为方法

① C程序设计语言的教学与实践，
奠定扎实的程序设计基础；

② 自顶向下、逐步求精的程序设计方法
及N-S图的计算过程描述，
培养良好的程序开发习惯；

③ 可编程结构模型
迭代、递归、穷举、回溯等算法，
初步具备抽象与程序设计的能力；

④ 结构体、枚举类型及数组、链表的学习
初步掌握“数据结构+算法=程序”
提高实际问题求解的方法和能力；

⑤ 计算机组成原理和操作系统原理的
初步了解和学习，
能够理解程序的执行过程及原理，
初步了解计算机系统

⑤ 介绍计算机发展过程中计算理论的作用，
计算机学科知识图谱和知识体系的教学，
提高对计算机科学与技术专业的认知，
明确专业学习的目标与方向。

教学思路（1）

基于计算机学科“理论-抽象-设计”3个学科形态，教学设计思路“感性→理性→实践”循序渐进、不断深化与螺旋上升的方法。

1、首先从编写C语言程序入手，对计算机程序语言初步认识和感知。

2、围绕几种典型算法设计，引入计算模型，结合程序语言学习，加强程序编程训练。

3、案例教学与实践，应用抽象与理论方法对问题进行分析，并应用程序设计求解问题。

教学思路（2）

4、围绕数据的存储，结合数组、结构类型的学习，引入数据结构的知识，初步理解“程序=算法+数据结构”

5、数据抽象与设计的案例教学，初步理解和掌握数据存储设计的方法

6、引入计算机组成结构，结合程序语言学习，初步理解计算机的组成及工作原理

7、基于学生对计算机系统认知基础，以程序在计算机中执行的过程为主线，通过语言编译、计算机工作原理、操作系统相关知识讲解，加深对计算机系统的认知和理解。

北京郵電大學

教学环境

本课程采用：

线下（课堂教学）+线上（OJ平台）实践

线上（QQ群）交流与互助

教学方法

1、掌握基本的程序设计方法，具备基本的程序设计能力，多练习多实践是关键：

“课上练、课后练、以考带练”

教学方法

利用各类OJ平台（包括学校OJ、PTA平台）

① OJ 平台上的课堂练习，调动学生课堂学习的参与度，及时了解学生的掌握情况、以便及时发现问题并调整讲授重点，增加师生之间的互动；

② 每周布置6~8个练习作业题，教师根据学生完成作业情况及时发现存在的问题，通过课堂讲解及时解决问题；

教学方法

2、在具备基本编程能力的基础上，针对理论课教学，老师主要讲授求解问题的**思路和方法**，特别是计算的数学模型。另外，加强复杂问题求解的**案例教学**和训练，同时在课后实践环节中也相应地增加复杂问题的求解练习。

讲义与教材

讲义与教材 （教材正在修订中）

1.《计算机导论与程序设计基础》

北京邮电大学出版社

-计算机工作原理、程序设计语言基础

2.《计算导论与程序设计实验指导书（第二版）》

北京邮电大学出版社

-编程实验

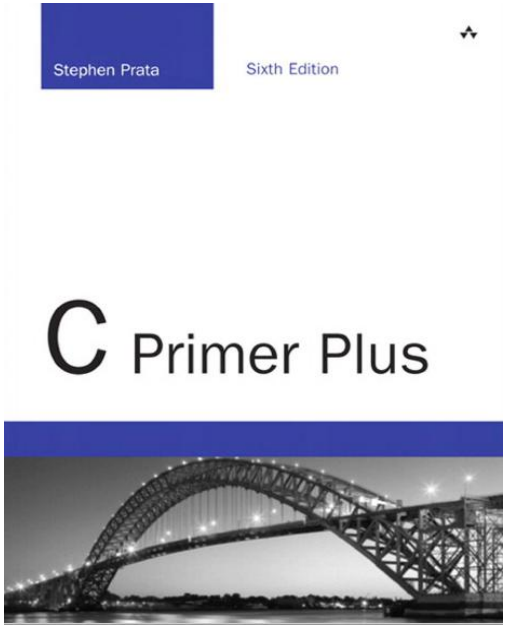
讲义与教材

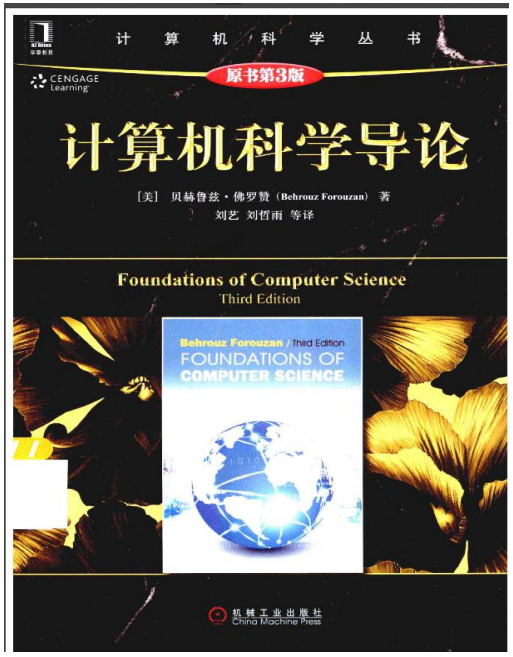
参考书：

3. 《CPrimer Plus(第6版)》中文版 人民邮电出版社

--C程序设计语言的详细讲解

4、《计算机科学导论（第3版）》中文版 机械工业出版社





北京郵電大學

教学内容

计算导论

- 程序设计语言
- 计算与计算模型（原始递归模型、自动机模型）
- 计算机硬件系统的组成结构和工作原理
- 计算机系统（软件系统、硬件系统）
- *计算机科学的数学基础（逻辑、集合关系、代数）
- 计算学科基本形态和知识体系

程序设计

- 程序（算法）的构造，设计方法
- C程序设计语言

- 课程学习注意点：
  - 课程授课重点是程序设计的方法和算法设计，学了语言不等于会编程;
  - 不仅要能设计好的算法，还需要培养良好的工程意识和工程规范，逐步培养专业性；
  - 要学好程序设计，必须要自己动手编程实践，多上机练习、多独立思考、多参与讨论；*“我听到的会忘掉，我看到的能记住，我做过的才真正明白！”*

- 课程学习注意点（续）：
  - 独立思考，学会使用程序调试方法去发现问题所在；和同学多讨论（*三人行，必有我师*）；善于利用互联网去寻找答案（*知识获取能力*）。
  - 扎实掌握每一个知识点。知识点是一环扣一环的，前面知识掌握不扎实会影响后续知识的学习。
  - 不必担心自己计算机操作基础薄弱，关键是从现在开始做起。

北京郵電大學

课程教学方式

1. 课堂授课（每周4节课，共15周）；
2. 上机作业在OJ平台（在线编程平台）完成并提交
3. 答疑（QQ群）

考核方式（计划）

平时成绩：作业

期中考试：笔试

期末考试：笔试

北京郵電大學

教学网站：北邮教学云平台

- **https://ucloud.bupt.edu.cn/**
- **用户名和初始密码均是大家的学号**
- **可以访问该网站下载讲义、作业**
- **可以利用该平台的论坛功能进行交流和答疑**

课程教学QQ群

群名称:2023.计导.国院.11~13.23~...
群　号:806540541

北京郵電大學

要学好本课程，要勤思考、多练习！

让我们一起努力，教好、学好！

遇到问题，请及时反馈

欢迎大家用邮件、QQ和我们沟通！

北京郵電大學

Unit 1: Programming language

Chapter 1: overview of programming language

北京郵電大學

Outline

1.1 What is programming language

1.2 The evolution of programming language

1.3 Build and run programs

1.4 Classification of programming languages

1.1 What is programming language

1. The concept of language:

- Language（语言）is the most important tool of human communication and the main way of **expression** for people to communicate.
- In a broad sense, language is a set of commonly used **communication symbols**, **expressions** and **processing rules**.
- Definition of language: a symbolic system composed of **words** and **grammar** and capable of **expressing human thoughts**.

Language: It is a tool for expression and description. Language is based on a set of characters (words) and a set of rules (grammar).
According to the rules, the whole string (sentence) composed of words is language.

1.1 What is programming language

2. The concept of computer language:

- **Computer language** （计算机语言）refers to the language used for communication between **people** and **computers**. Computer language is the medium of transferring information between people and computers.
- In order for the electronic computer to perform various tasks, it is necessary to have a set of numbers, characters and syntax rules for writing **computer programs**, and these characters and syntax rules constitute various **instructions** (or various statements) of the computer. These are the languages that computers can accept.
- The biggest characteristic of computer system is that **instructions** are transmitted to the machine **through a language**.

1.1 What is programming language

3. The concept of programming language

Refer to the procedures in daily work

- **Program:** Sequence of instructions (operations).
  - The instruction sequence with special functions, arranged in advance according to the work steps.
- **Programming:** Programming is the process of giving programs to solve specific problems, and is an important part of program construction activities. The programming language is often used as a tool to give the program in this language.

Problem space → Solution (construction) process → Solution space (C language program)

- **Programming language:** A language used to write computer programs to express and describe the data to be processed and the steps and processes for solving problems. It is an overall character string composed of characters on a limited alphabet according to predefined rules (morphology and syntax).

Example: program and programming language

```c
#include <stdio.h>

int main(void)
{
    printf("Hello, world!");
    return 0;
}
```

words: include, main, return, printf ...

Statement (sentence): return 0;

Note: Sentences are strings ending with ";".

Syntax (rules) : {Statement[Statement]}

Example: program and programming language

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

main()
{
    int number1;   /* Variable declaration */
    int number2;
    int result;

    scanf(“%d%d”,&number1,&number2); /*Input, read 2 integers from the keyboard */

    result = number1 * number2;    /* Multiplication */

    printf（“the result is : %d\n ”, result）; /* Output results */

    return 0;
}
```

This is a program written in C language, which describes the multiplication process of two numbers
Operation sequence: Read in two numbers -> Multiplication -> Output results

According to predefined rules (Syntax), the total number of strings composed of characters in a limited alphabet

Example: program and programming language

1	00000000	00000100		0000000000000000
2	01011110	00001100	11000010	0000000000000010
3		11101111	00010110	0000000000000101
4		11101111	10011110	0000000000001011
5	11111000	101011101	11011111	0000000000010010
6		01100010	11011111	0000000000010101
7	11101111	00000010	11111011	0000000000010111
8	11110100	10101101	11011111	0000000000011110
9	00000011	10100010	11011111	0000000000100001
10	11101111	00000010	11111011	0000000000100100
11	01111110	11110100	10101101	
12	11111000	101011110	11000101	0000000000101011
13	00000110	10100010	11111011	0000000000110001
14	11101111	00000010	11111011	0000000000110100
15		00000100	00000100	0000000000111101
16		00000100	00000100	0000000000111101

program: The instruction sequence with special functions arranged in advance according to the work steps

北京郵電大學

Outline

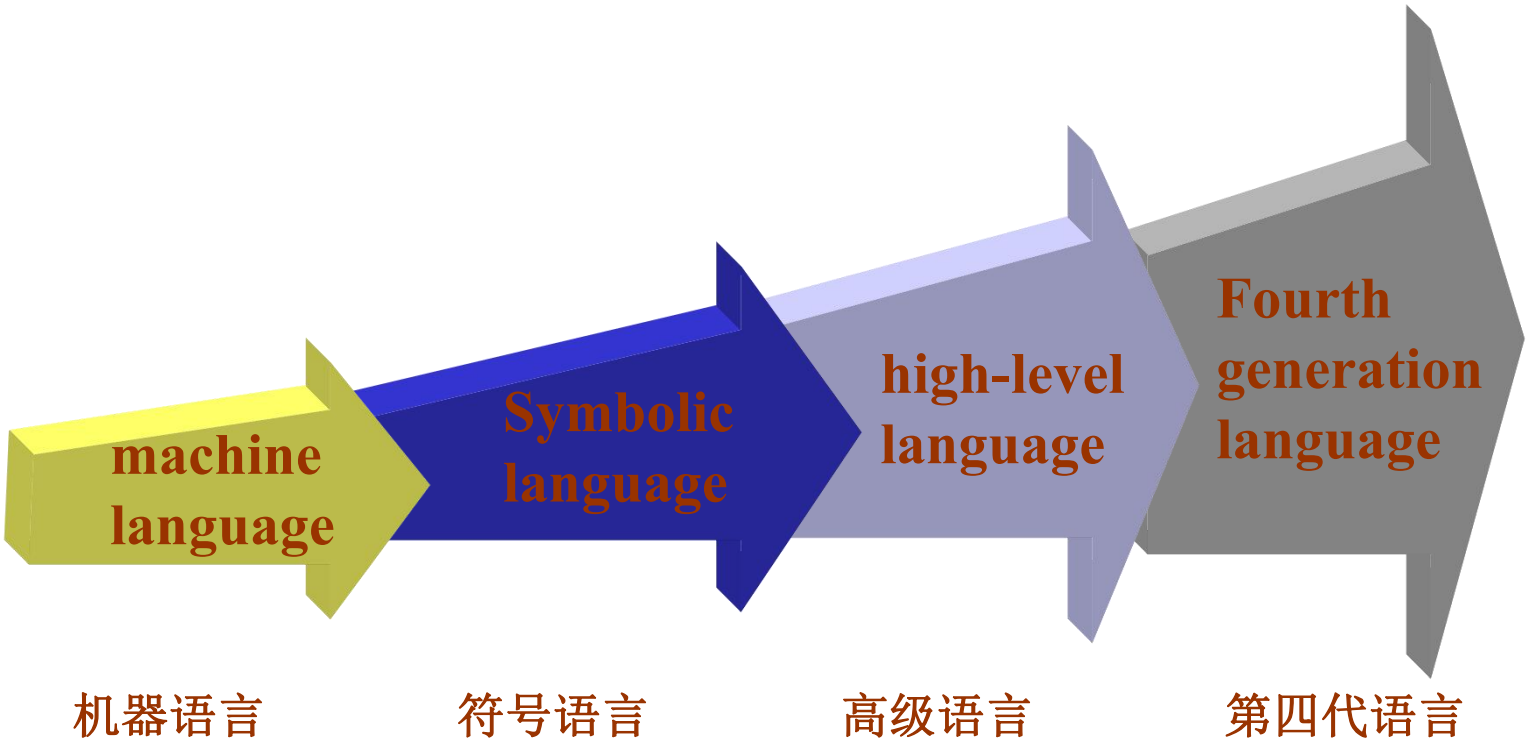
1.1 What is programming language

1.2 The evolution of programming language

1.3 Build and run programs

1.4 Classification of programming languages

1.2 The evolution of programming language



1.2 The evolution of programming language

1. Machine language

- Languages used in the early days of computer development;

Refers to all the instructions of a computer.

- The instruction is composed of binary codes of "0" and "1 ".

It is the command to instruct the computer to work, and it is the only language that can be directly recognized by the computer;

- Depending on the specific machine, different machines can recognize different machine languages;

1.2 The evolution of programming language - machine language

1	00000000	00000100		0000000000000000
2	01011110	00001100	11000010	0000000000000010
3		11101111	00010110	0000000000000101
4		11101111	10011110	0000000000001011
5	11111000	101011101	11011111	0000000000010010
6		01100010	11011111	0000000000010101
7	11101111	00000010	11111011	0000000000010111
8	11110100	10101101	11011111	0000000000011110
9	00000011	10100010	11011111	0000000000100001
10	11101111	00000010	11111011	0000000000100100
11	01111110	11110100	10101101	
12	11111000	101011110	11000101	0000000000101011
13	00000110	10100010	11111011	0000000000110001
14	11101111	00000010	11111011	0000000000110100
15		00000100	00000100	0000000000111101
16		00000100	00000100	0000000000111101

1.2 The evolution of programming language - machine language

The flaws of machine language：

- Very obscure and difficult to read;
- The writing workload is large, and it is easy to make mistakes and difficult to modify;
- Related to a specific machine
  - developers are required to have a very correct and in-depth understanding of the computer's hardware and instruction system, as well as proficient programming skills, so only a few experts can meet this requirement;
- Poor portability(可移植性) (machine language programs written on one machine may not run on another machine of a different model).

1.2 The evolution of programming language - symbolic language

2. Symbolic language

- In the early 1950s, mathematician Grace Hopper invented symbolic languages, which use symbols or mnemonics（助记符） to represent different machine language instructions (including opcodes and operand addresses).
- Programmers can use symbols such as ADD, SUB, MUL, and DIV to represent opcodes for addition, subtraction, multiplication, and division, respectively.
- Symbolic language is also called assembly language.

Assembly language program to find $d = b^2 - 4ac$

opcodes

operand address

	Program		Comment (operation result)
1.	MUL B B	;	put b^2 into B
2.	MUL A E	;	put 4a into A
3.	MUL A C	;	put 4ac into A
4.	SUB B A	;	put $b^2 - 4ac$ into B
5.	MOV D B	;	move $b^2 - 4ac$ from B to D
6.	HLT	;	stop

A	a
B	b
C	c
D	d
E	4

Use symbols or mnemonics to represent opcodes and operand addresses in instructions

1~6 are instructions, MUL is a multiply instruction, SUB is a subtraction instruction, MOV is a transfer instruction, and HLT is a stop instruction;

A, B, C, D, and E represent registers that store the numbers a, b, c, d and the constant 4, respectively.

1.2 The evolution of programming language - symbolic language

- A program written in assembly language needs to be translated into machine language (binary code) to run, and this translation process is implemented by an assembly program.

1.2 The evolution of programming language - symbolic language

The limitations of assembly language:

- The syntax and semantic structure of assembly language are still basically the same as those of machine language, and are far from the traditional method of human to solve problems.
- Most of the instructions in assembly language are in one-to-one correspondence with machine instructions, so the amount of code is large.
- Related to specific machines, people still have to have a correct and in-depth understanding of the computer's hardware and instruction system, and they still have to memorize machine language symbols (mnemonics). Portability is not good.

1.2 The evolution of programming language - high-level language

3. High-level language

- Due to the limitations of assembly language, high-level languages appeared later.
- High-level languages are very similar to natural languages (especially English), so high-level language programs are easy to learn, understand, and troubleshoot.

C program to multiply two numbers

```
#include <stdio.h>
main()
 {
    int number1;
    int number2;
    int result;
    printf("please input the two numbers:");
    scanf("%d %d",&number1,&number2); // Read multiplier
and multiplicand
    result = number1 * number2; // multiply two numbers
    printf ("the result is : %d\n ", result) ;//output
the result
    return 0;
}
```

书上31页错误，将16行和18行的d%改成%d

1.2 The evolution of programming language - high-level language

The advantages of high-level languages

- High-level language programs are easy to learn, understand, and troubleshoot.
- Programmers can completely do not have to deal with the hardware of the computer, do not have to understand the instruction system of the machine.
- High-level languages have nothing to do with specific machines, and high-level language programs running on one machine may be transplanted to run on another machine without modification, which greatly improves the versatility of the program.

1.2 The evolution of programming language

4. Fourth-generation language

Fourth-Generation Language (hereinafter referred to as 4GL) is a problem-oriented programming language that realizes **abstraction at a higher level**, which can greatly improve software productivity and shorten software development cycles.

4GL provides a powerful non-procedural problem definition method. The user only needs **to tell the system what to do** without explaining how to do it, and the program can automatically generate an algorithm and process it automatically.

Typical 4GL languages are ADA, MODULA-2, SMALLTALK-80, etc.

1.2 The evolution of programming language

According to the function of 4GL, it can be divided into the following categories:

- Query language: It is the main tool of the database management system, which provides users with the function of querying the database. Such as SQL (Structured Query Language);
- Report Generator: It is an important tool for users to automatically generate reports. It provides non-procedural description means for users to easily generate reports based on the information in the database, such as ADF.
- Graphical language: software development in a graphical way.
- Application Builder: It is an important type of comprehensive 4GL tool, which is used to generate a complete application system. Application Builder frees users from having to use multiple pieces of software, but instead uses such a comprehensive tool to implement multiple functions.
- Formal specification language: A formal statement of the functions, performance, and other important aspects that software should meet, which avoids the ambiguity of natural language and is the basis for software automation.

1.2 The evolution of programming language

5. Natural language

Ideally, the computer can understand natural language (e.g. English, Chinese, etc.) and execute the request immediately.

A lot of work on natural language is going on in the lab.

But so far, the use of natural language is still quite limited.

Jim was happy in the morning, however he broke his lovely toy and was sad all day。

Question: is Jim happy or not?

Summary

Programming language:

- The language used to write computer programs that **express and describe** the data to be processed and the steps and processes for solving problems.
- A collection of **strings (words and sentences)** composed of characters from a finite alphabet according to predefined **rules (grammar)**.

北京郵電大學

Outline

1.1 What is programming language

1.2 The evolution of programming language

1.3 Build and run programs

1.4 Classification of programming languages

1.3 Build and run programs

1.Where is the program written? Where is it stored? --- file as carrier

- A high-level language program (C program) is a collection of strings, and characters are text, similar to the English text we often use;
- A program written in C language can be written and saved with a text editor (TXT editor, etc.); we call it "**source program**";
- Since the computer can only execute binary-coded instructions, programs written in C language cannot be directly executed on the computer.
- We need to "transform" the program written in C language into a machine instruction sequence program, which we call "**object program or executable program**"

1.3 Build and run programs

2. Translator---compiler and interpreter

- Translation of high-level language programs into machine language programs requires the help of translation programs.
- Translation is actually a "transformation" process, that is: a computational process
- The translation process is carried out according to the grammatical rules of the programming language, so grammatical or lexical errors can be checked at the same time;
- The translator is also a program, provided by a third party, we just use
- There are two kinds of translation programs: **compiler** and **interpreter**.

1.3 Build and run programs

- Compiler（编译程序）:
  - Translate all the statements in the source program into a machine language program **at one time**, and then run the machine language program.
  - Compiling and running are two separate phases. If you want to run the same program multiple times, as long as the source program remains unchanged, you do not need to recompile; if the source program is modified, you need to recompile.

1.3 Build and run programs

- Interpreter（解释程序）：
  - Translate a statement in the source program into machine language and execute it immediately (and no longer save the machine language program just executed), and then translate and execute the next statement. Repeat this until the program ends.
  - If a statement is to be repeatedly executed, the statement must be retranslated for each repeated execution, so the efficiency is very low.

1.3 Build and run programs

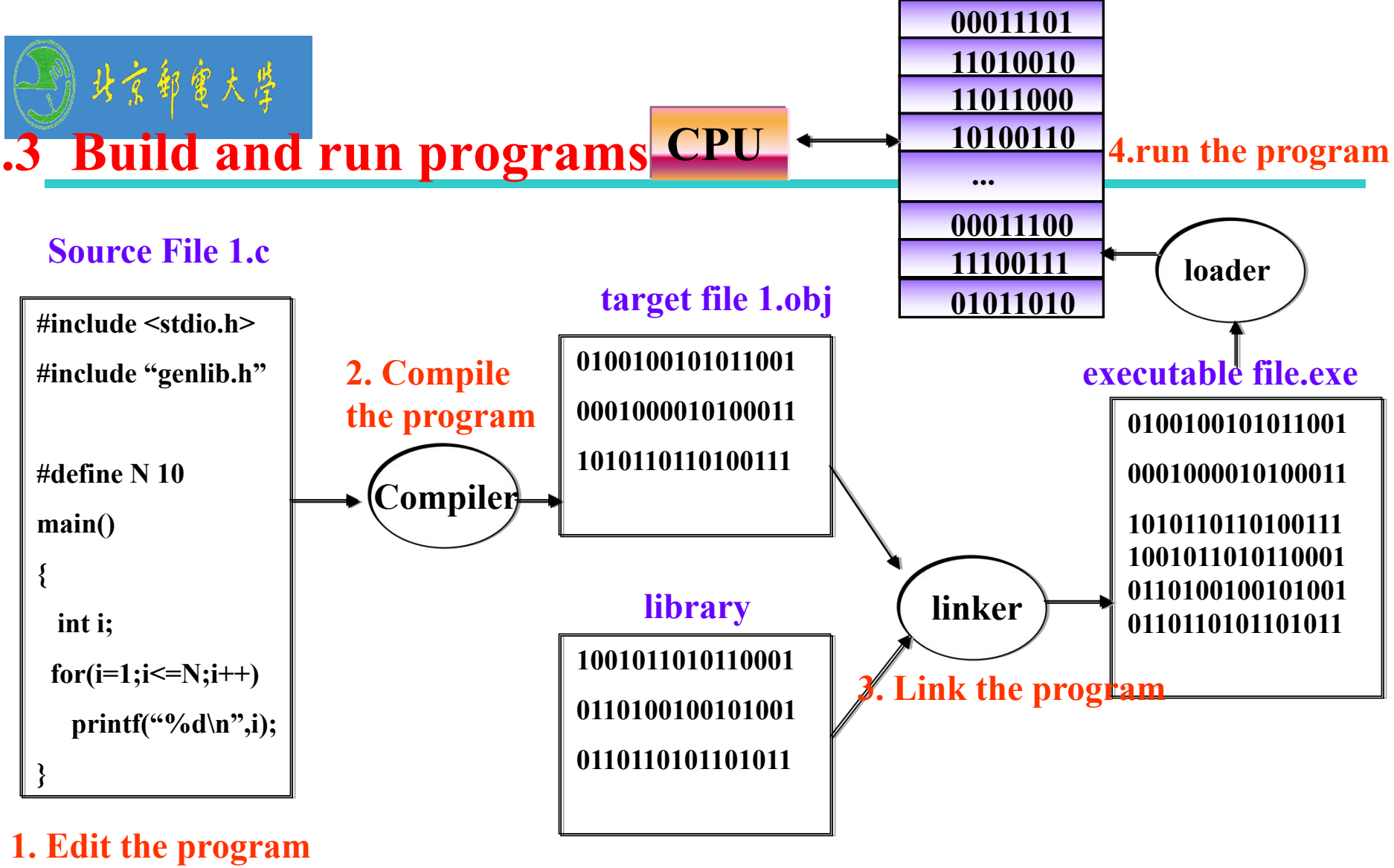
- Famous interpreters include:
  - BASIC language interpreter, LISP language interpreter, UNIX command language (shell) interpreter, database query language SQL interpreter, etc.
- At present, most languages such as C, C++, FORTRAN, ALGOL, etc. are translated by compilers.
- BASIC, PASCAL, LISP, etc. have both compiler and interpreter.

1.3 Build and run programs

3.linker

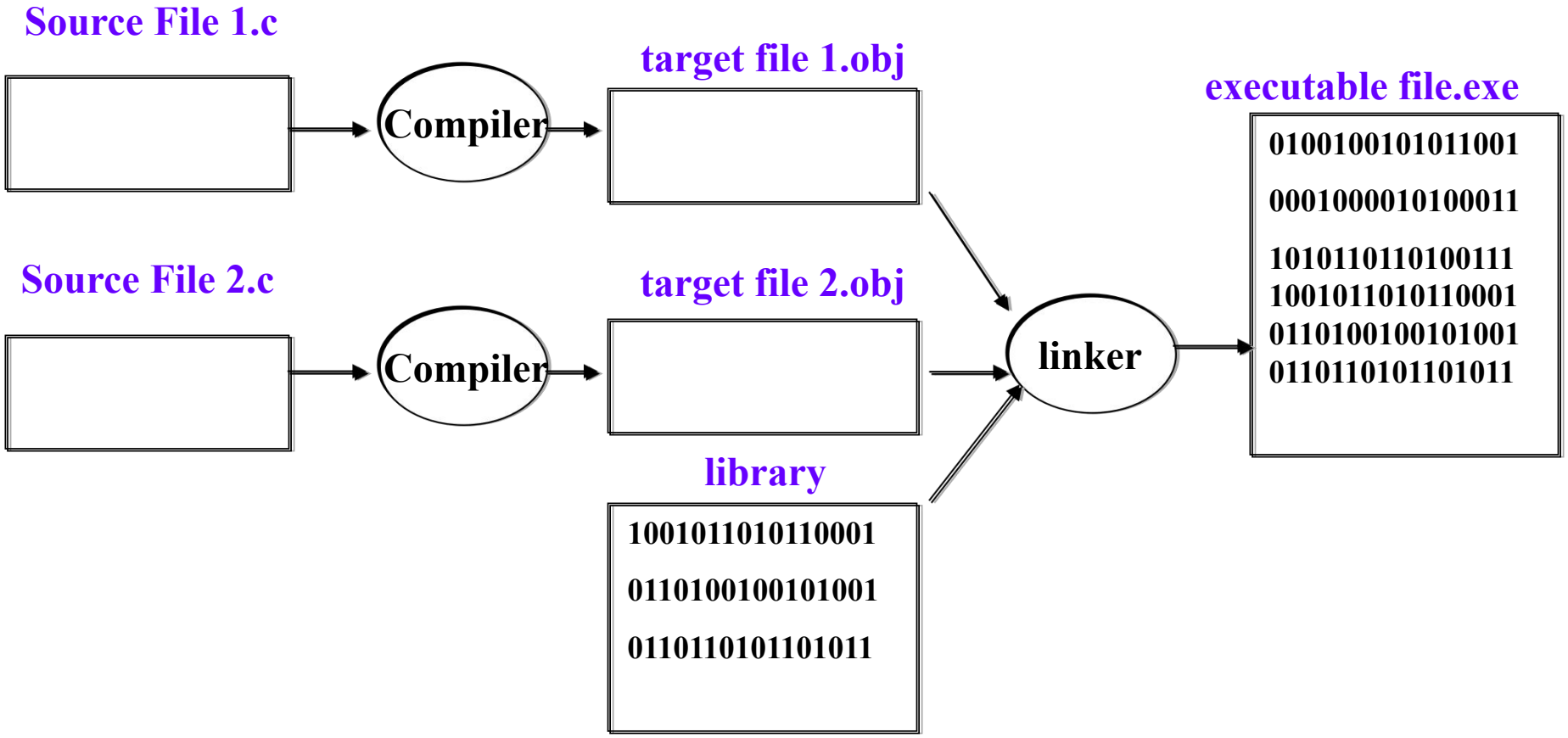
- There is a distinction between **object code**(the machine code that has not yet been linked) and **executable machine code**.
- Linker collects code compiled or assembled separately in different object files (.obj) into a single directly executable file (.exe).
- The linker also links the object program with the code for the **standard library functions**, and links the object program with the **resources provided by the computer's operating system** (e.g., storage allocators and input and output devices).
- The linking process extremely dependent on the operating system and the processor.

1.3 Build and run programs



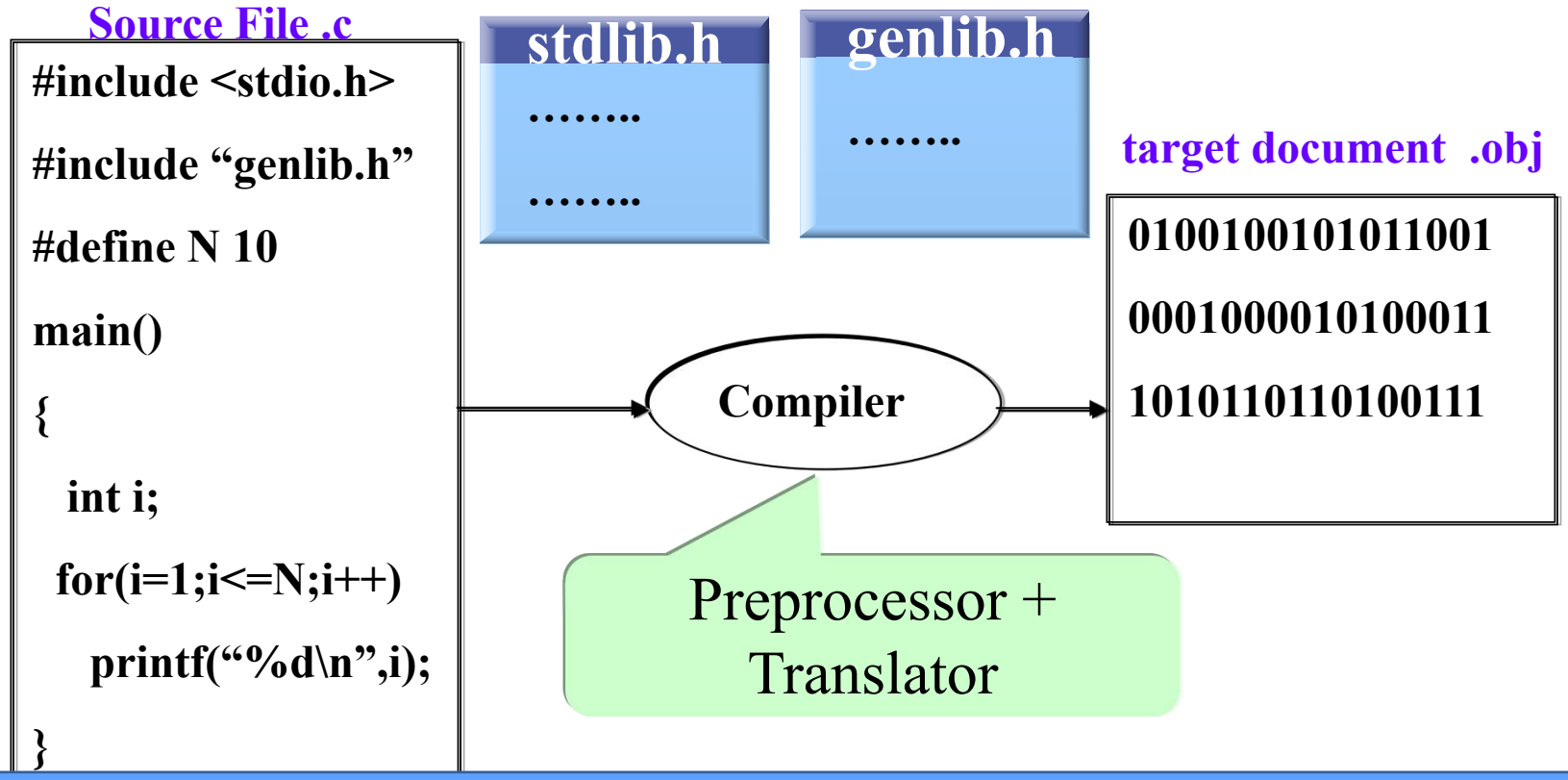
C Standard Library: There is a rich set of functions that can be used directly by programmers.

1.3 Build and run programs



When a program consists of multiple source files

1.3 Build and run programs



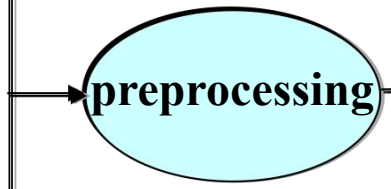
Preprocessor: Usually includes other files into the file to be compiled, and replaces special symbols with program text. All preprocessor instructions in C language begin with #.

Translator: Translate the preprocessed program into binary code.

1.3 Build and run programs

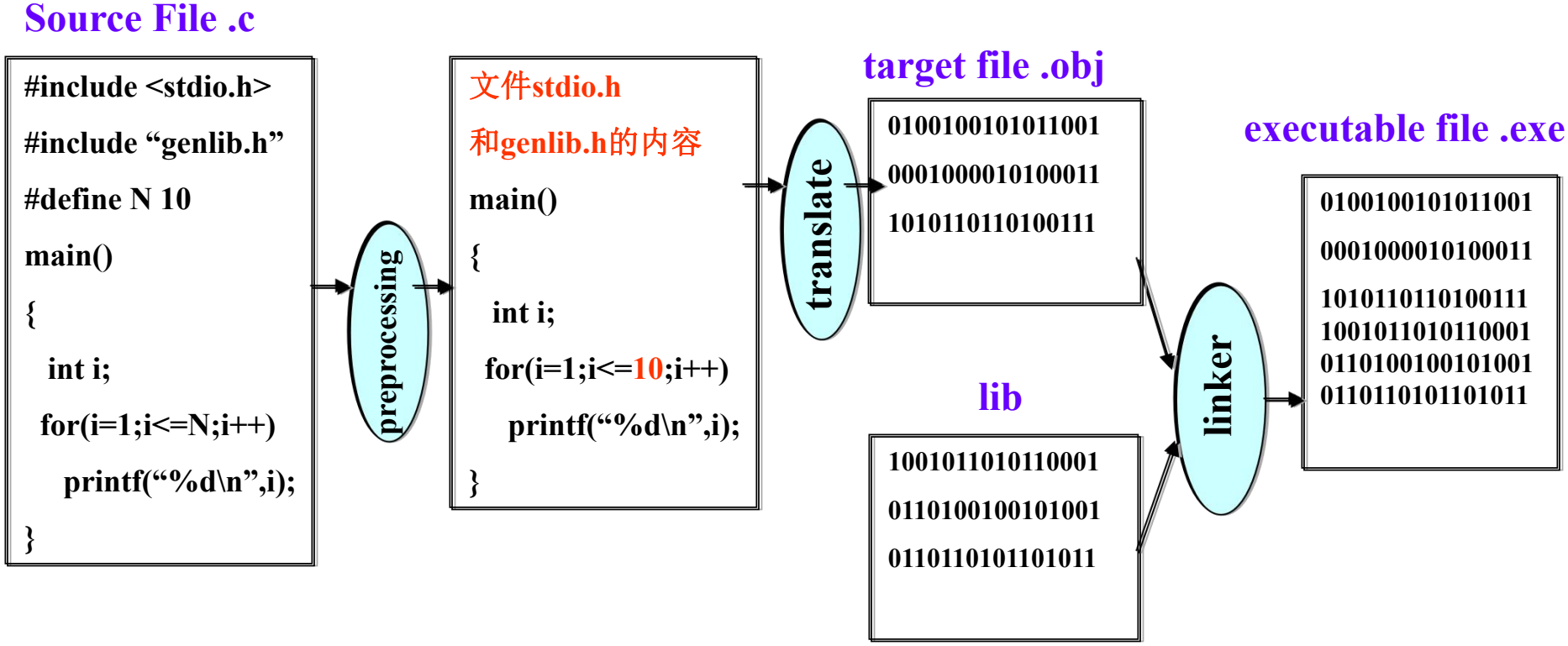
Source File.c

```
#include <stdio.h>
#include “genlib.h”
#define N 10
main()
{
   int i;
  for(i=1;i<=N;i++)
     printf(“%d\n”,i);
}
```




```
文件stdio.h
和genlib.h的内容
main()
{
   int i;
  for(i=1;i<=10;i++)
     printf(“%d\n”,i);
}
```

1.3 Build and run programs

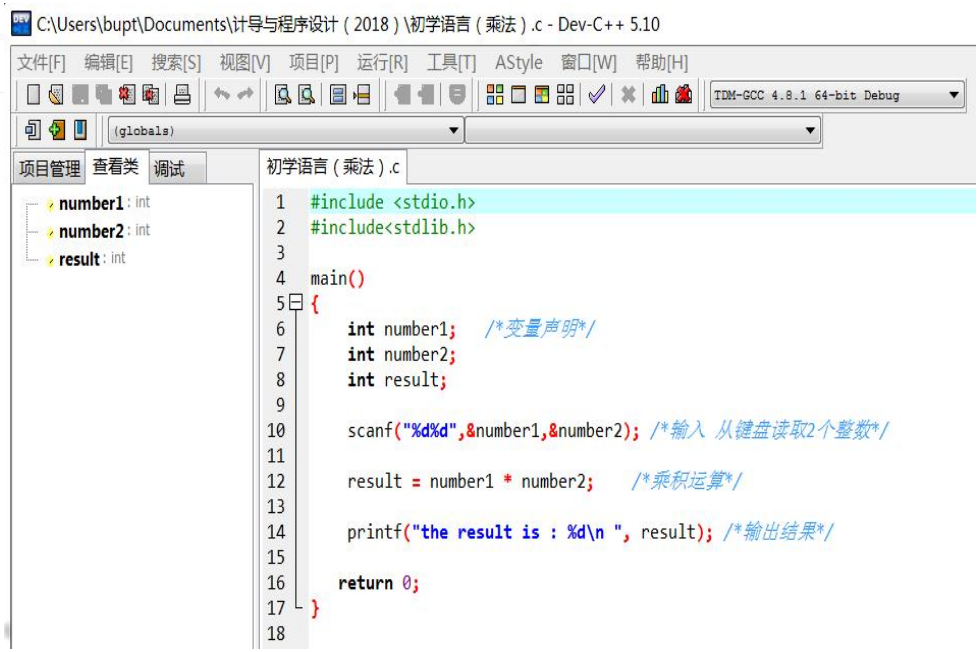


1.3 Build and run programs

Introduction to the Integrated Development Environment

- DevCpp

https://pan.baidu.com/s/1rKEZ1MlZv7bys_iB5M2i1Q



1.3 Build and run programs

Demonstrate how to build and run a C program in the DEV C++ integrated development environment

The basic steps:

1.Source programs are created and written using the "File" menu function, and stored as ".c" files.

2.Use the **compile function** in the "Run" menu to compile the source program written. Syntax check is performed during compilation, and if there are syntax errors, they must be corrected.

3.You can run the program only after the compilation result has no error warning, and use the **run function** in the "Run" menu. See the results of the program running.

1.3 Build and run programs

Thinking:

Suppose there is such a statement in a program: a=b+c; its storage content on the hard disk is:**01100001 00111101 01100010 00101011 01100011**

Since the program code is already binary, why can't it be executed directly? Why do we need to compile it again?

The compiled binary code looks like:

move register R1 the memory address of b
move register R2 the memory address of c
add R1 R2
move the memory address of a R1

1.3 Build and run programs

Extended thinking:

The compilation of a program can be regarded as a "calculation" process, which is calculated according to the grammatical rules of the programming language. So **how should the grammatical rules be described**?

Reference answer:

The way people describe things with a set of **mathematical symbols** and **rules** is called a **formal description**, and the mathematical symbols and rules used are called **formal language.**

Formal Languages and Automata studies the theory of formal definition, classification and structure of programming languages and natural languages, and identify the formal models of various languages (automata models) and their relationships.

1.3 Build and run programs

✧ **Example：** **Assume alphabet(字母表) T = {a，b}**

则 L_1 = {$a^n b^n$ ｜ n≥1}

L_2 = { b^k ｜ k is a prime number(质数)}

- **L_1 、L_2** are both languages (symbolizations, rules, strings)
- Mathematical Methods: Set Operations.

北京郵電大學

Outline

1.1 What is programming language

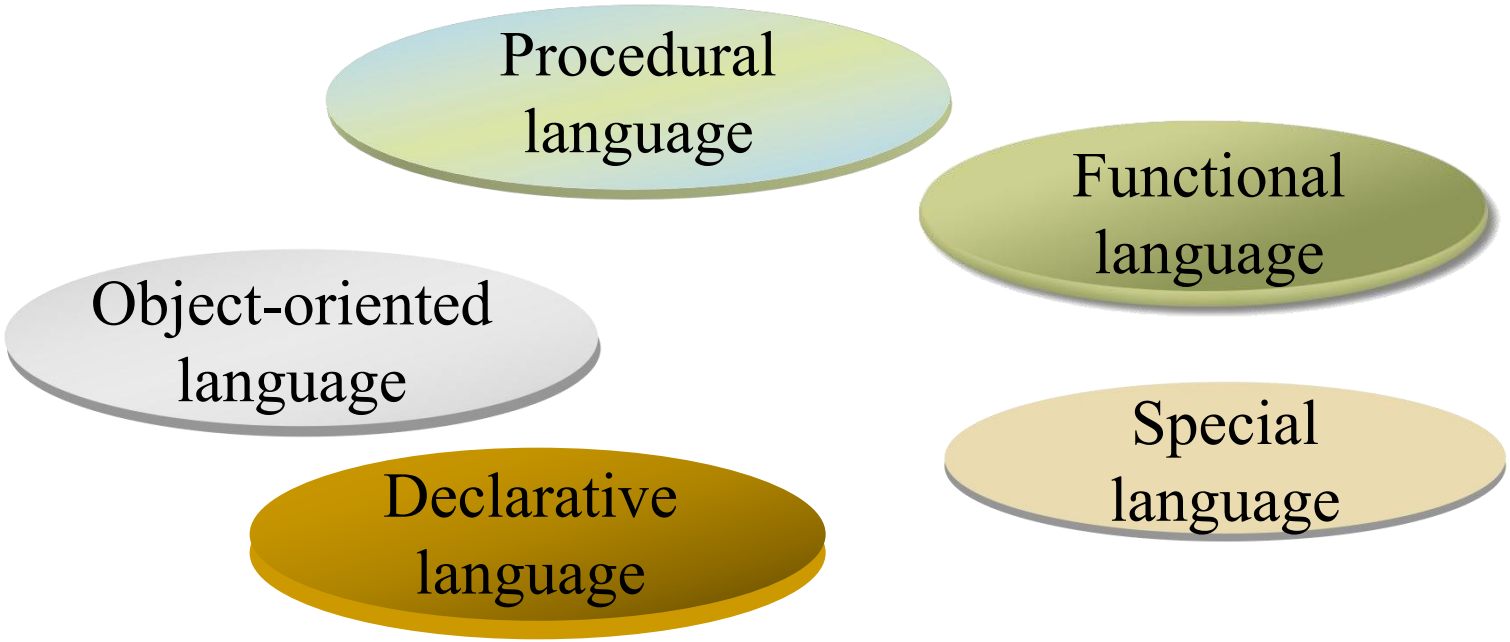
1.2 The evolution of programming language

1.3 Build and run programs

1.4 *Classification of programming languages

1.4 Classification of programming languages

- Programming languages can be categorized according to the way people solve problems:



1.4 Classification of programming languages

1. Procedural language

A language for describing computational processes based on von Neumann structures.

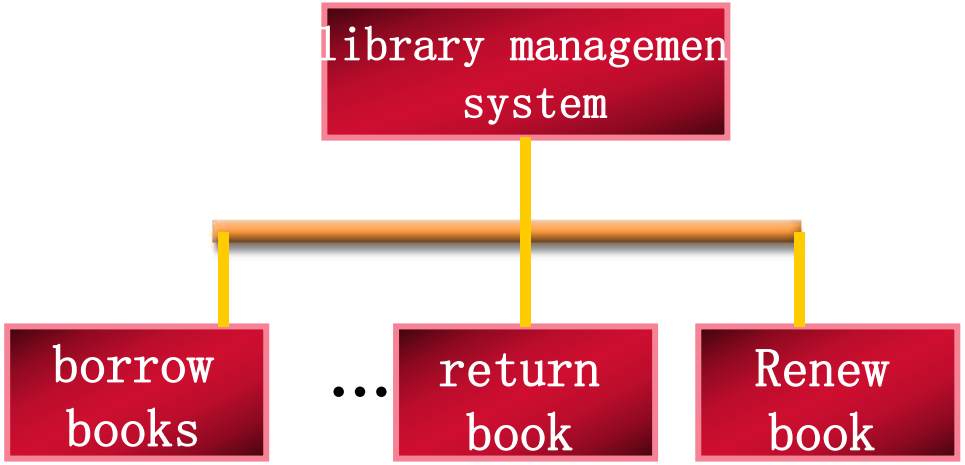
Process-oriented programming features:

- **Action(activity)-oriented** . Solving a problem (program) is a complex computational process that can be decomposed into several actions (activities). That is: a calculation process can be regarded as a series of actions.
- **An action** can be a statement (basic operation) or a sub-procedure (subroutine) that can continue to be decomposed.

1.4 Classification of programming languages

- **Sub-procedure (subroutines)**: Each subroutine implements a distinct, independent function. Applying the above method continues to decompose into a series of actions.
- The cycle repeats until the "action" is decomposed into "basic operation-statement".
- In a word, process-oriented design is to realize the operation process and operation steps of a certain calculation, and then use procedural language to describe these operation processes and steps.

1.4 Classification of programming languages



The library management system is divided into functions such as borrowing books, returning books, and renewing books. The realization of each function corresponds to an execution process, and there are a series of execution steps.

A simple borrowing process might look like this:

1. Read the borrower's library card information to verify the legitimacy of the borrower's identity;
2. If the identity is legal, verify whether the borrower has unpaid debts;
3. If there is arrears, record the payment information of the borrower;
4. Verify whether the borrower have reached the allowed number ;
5. If the book can be borrowed, the borrower's borrowing information will be recorded in the system;

...

Operation process described in procedural language

1.4 Classification of programming languages

FORTRAN	**The first high-level language, designed by a group of IBM engineers under the leadership of Jack Backus, was commercially available in 1957. Suitable for scientific computing.**
COBOL	**A language for business data processing.**
PASCAL	**1971, invented by Niklaus Wirth in Zurich, Switzerland. Teach beginner programming by emphasizing structured programming methods. Suitable for research and teaching, but never gained widespread popularity in industry.**
C	**Invented by Dennis Ritchie in the early 1970s, it was originally intended for use in writing operating systems and system software. It has advanced data structures and control structures, and can also call the underlying functions, with short and efficient instructions.**
ADA	**Developed for the U.S. Department of Defense, a distributed systems control language.**

1.4 Classification of programming languages

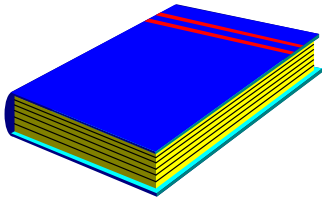
2. Object-oriented language

The basic idea of object-oriented:

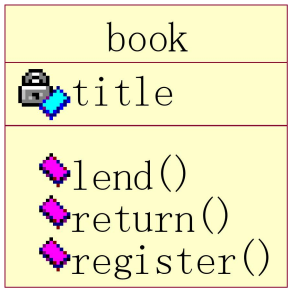
- The software system is constructed from the objective things in the real world, and it is emphasized that the things in the problem domain (the real world) are directly taken as the center to think and understand the problems, .According to the essential characteristics of these things, they are abstractly expressed as objects in the system, as the basic unit of the system. (object)
- Objects contain properties and operations, and each object has clear responsibilities and completes certain functions. (attributes and operations)
- Objects are not isolated, but have various relationships.
- Objects communicate with each other through messages, and cooperate with each other. (message)
- Objects of the same type can be further abstracted from commonalities to form classes. (class)

1.4 Classification of programming languages - object-oriented language

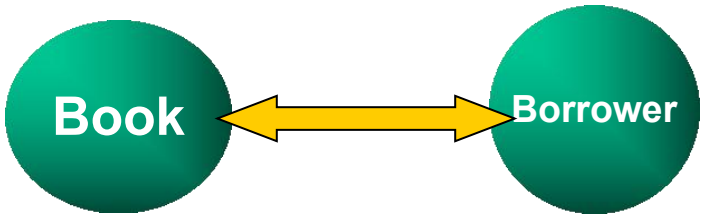
- Object-oriented languages can be used to describe the objects involved in problem solving and the relationships between objects.



abstract



Java Description



The 2 main classes in the library management system: Book, Borrower

```
Public class Book
{
  Private String title;
  Public Boolean lend(){
      ……
  }
  Public Boolean return(){
      …
  }
}
```

1.4 Classification of programming languages

Two major object-oriented languages

C++	Extended C language with object-oriented features.
JAVA	Pure object-oriented language. Developed by Sun Corporation, it is developed on the basis of C and C++, but some language features in C++ are removed to make it stronger.

1.4 Classification of programming languages

3. Functional specification language

The idea of solving problems based on mathematical modeling.

The basic unit of a program is a function. A function can be understood as a black box that maps from a series of inputs (X) to outputs (Y).

Functional languages mainly implement the following functions:

- Define a series of basic functions that can be called by any programmer;
- Allows programmers to create new functions by combining several basic functions.

1.4 Classification of programming languages - functional specification language

Example:

define a basic function called **first**. It is used to extract the first element from a list.

Define another function **rest**, which completes the extraction of all elements except the first element from a list.

By combining the two functions, a function can be defined in the program to complete the extraction of the third element.

1.4 Classification of programming languages - functional specification language

LISP	**It was designed and developed by a research group at the Massachusetts Institute of Technology (MIT) in the early 1960s. Treat everything as a list, treat the list as a processing object.**
Scheme	**A series of basic functions are defined. The function name and the function's input list are written inside parentheses, and the result is a list that can be used as input to other functions. (car 2 3 7 8 11 17 20) == => 2 (cdr 2 3 7 8 11 17 20) == => 3 7 8 11 17 20 Now you can combine these two functions to get the third element 7 from the list: (car (cdr(cdr list)))**

1.4 Classification of programming languages - declarative (logical) language

4. Declarative (logical) language

Based on some assertions (facts) that are known to be correct, new assertions (facts) are derived using reliable principles of logical reasoning.

`If (A is B) and (B is C), then (A is C)`

Apply this guideline to the following facts:

fact1：`Socrates is a human →A is B`

fact2：`A human is mortal →B is C`

Then the following facts can be deduced:

fact 3：Socrates is mortal →A is C

1.4 Classification of programming languages - declarative (logical) language

- Programmers need to know all the known facts in the field, and should also be proficient in how to define rules logically rigorously so that programs can derive and generate new facts.
- Programs are domain-specific due to the large amount of factual information to be collected. Prescriptive programming has so far been confined to the field of artificial intelligence.

1.4 Classification of programming languages - declarative (logical) language

PROLOG： A declarative language for expert systems for artificial intelligence

Known Facts:

human(John)

mortal(human)

Users can make inquiries

?-mortal (John)

Program response is (yes)。

1.4 Classification of programming languages

5. Special language

- **HTML：** Hypertext Markup Language (HTML), a standard published by the World Wide Web Consortium (W3C), is specially designed for publishing information on the Internet.

- **HTML** files consist of text and tags (like <head>, <table>, etc.). Tags are pre-specified and defined, and users cannot create them themselves.

- Stored on the server side, it can be downloaded by the browser to the local for analysis and display.

1.4 Classification of programming languages - HTML

An **HTML** file consists of a header and a body.

header

text

tag

body

```
<HTML>
  <HEAD>
    <TITLE> 计算机科学导论与程序设计</TITLE>
  </HEAD>
  <BODY>
    This is the picture of a book:
    <IMG SRC= " P7081712.jpg "    ALIGN=MIDDLE>
  </BODY>
</HTML>
```

File header（文件头）: Contains the page title and other parameters that the browser will use.

Text is the actual information contained in the page. The markup defines the format in which the document is presented.

北京郵電大學

Brief Rewiew & Homework

- What is program, programming and programming language
- The process of compiling a computer source program (.c) into an executable program (.exe)
- The characteristics of machine language, symbolic language, and high-level language
- Homework
  - Install Dev c++
  - Read chapter 2 of 《计算机导论与程序设计》
  - Read chapter 1 of C Primer Plus