第1章 绪论

本章介绍课程信息、考核方式等，讲解编程基础概念、C语言的发展历史和特点、C语言快速入门基本概念等。

2024秋学期

A programmer joke

- A programmer is going to the grocery store and his wife tells him, "Buy a gallon of milk, and if there are eggs, buy a dozen." So the programmer goes, buys everything, and drives back to his house. Upon arrival, his wife angrily asks him, "Why did you get 12 gallons of milk?" The programmer says, "There were eggs!"

- 你找个有西瓜和西红柿的菜市场，告诉他，去买一个西瓜，如果看见西红柿，就买两个。要是买回一个西瓜和两个西红柿，他就不是真正的程序员，真正的程序员会买回来两个西瓜。

Why should I study C?

- There are many other languages like C++, JAVA, Python, etc .., then why should I study C?
- Learning C++ or Java directly is not a easy task. Learning the basic language elements makes your journey to Object Oriented languages a lot easier.
- No other language can beat C when it comes to PERFORMANCE. Major parts of operating systems like Windows, Linux, Unix are written in C.

主要内容

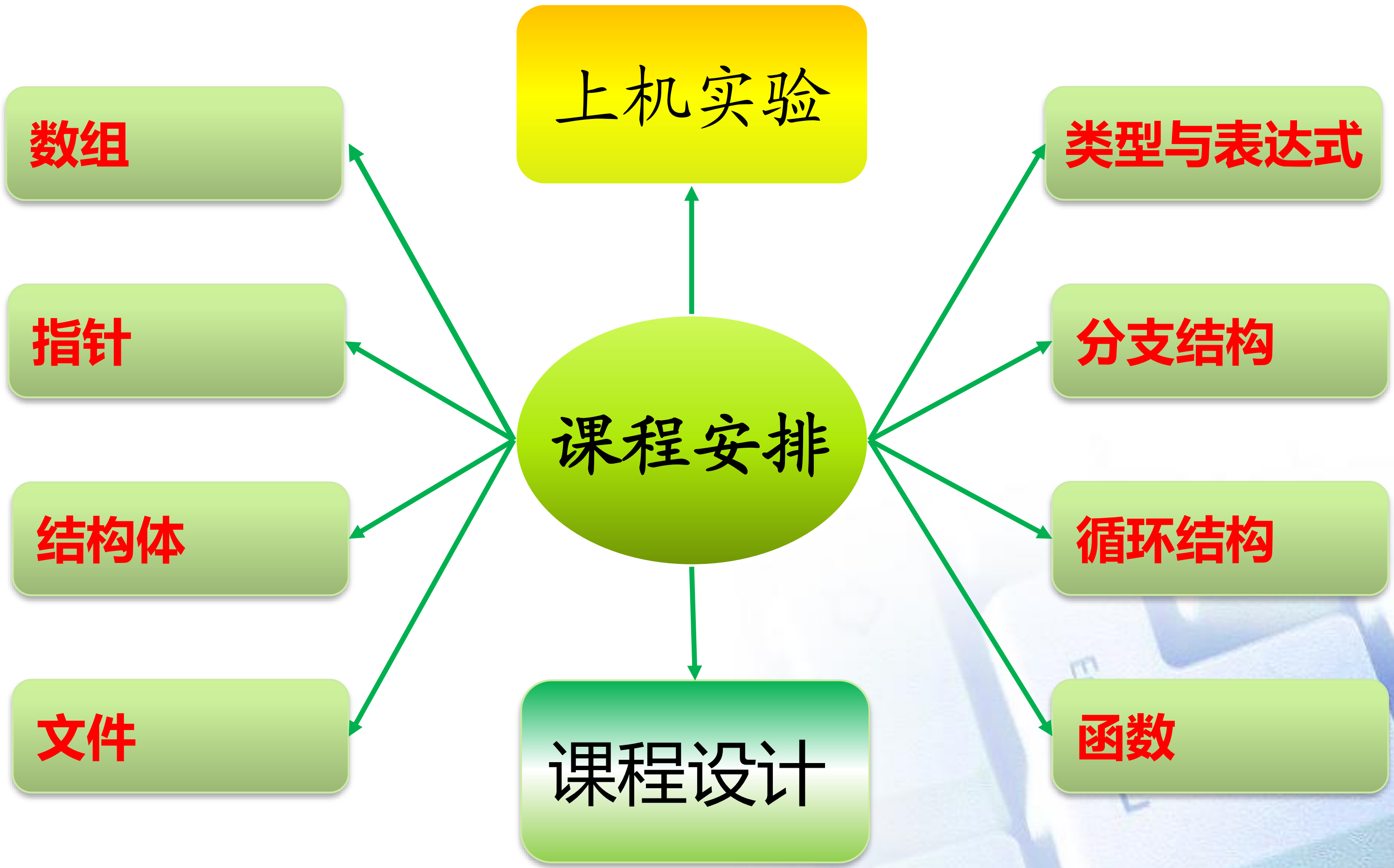
课程信息

- **教师:** 郜帅 教授
  - 邮箱：shgao@bjtu.edu.cn
  - 办公室: 机械工程楼D807
  - 电话: 51688743
- **课程资源:**
  - 交大课程平台：课程信息、课件资源等。
  - 课程实验：希冀平台：course.educg.net
- **课程教材:**
  - 教材：郑晓健. C语言程序设计（基于CDIO思想）. 第2版. 清华大学出版社.2017.7
  - 参考书：童晶. C语课程设计与游戏开发实践教程. 第1版. 清华大学出版社. 2020

课程目标

- 1、掌握C语言的基本语法知识；
- 2、掌握一种C语言开发环境，包括使用开发环境编写和调试程序的基本方法；
- 3、掌握阅读程序和分析程序的方法和技巧，掌握程序设计过程，**具备较熟练的程序编写和调试能力**；
- 4、学会针对实际应用问题，进行分析和抽象，并逐步分解，建立解题模型，通过程序设计训练，**提高解决实际问题的能力**。
- 5、**具备主动学习的能力**，能完成课下自学内容并通过考核，能在完成大作业过程中自学相关知识并顺利完成任务。

课程安排



课程安排

周次	1	2	3	4	5	6	7	8	9	10	11	12	13
星期一	🟨 绪论	🟩	🟩	🟩	🟩	🟩	🟨 期中复习	🟩	🟩	🟩	🟩	🟩	
星期三	🟨 数据类型	🟨 分支结构	🟨 循环结构	🟨 数组1	🟨 数组2	🟨 函数1	🟨 函数2	🟨 结构体	🟨 指针1	🟨 指针2	🟨 文件	🟥 课堂汇报	
星期六							🟥 期中考试						🟥 期末考试

黄色：课堂授课

绿色：课堂实验

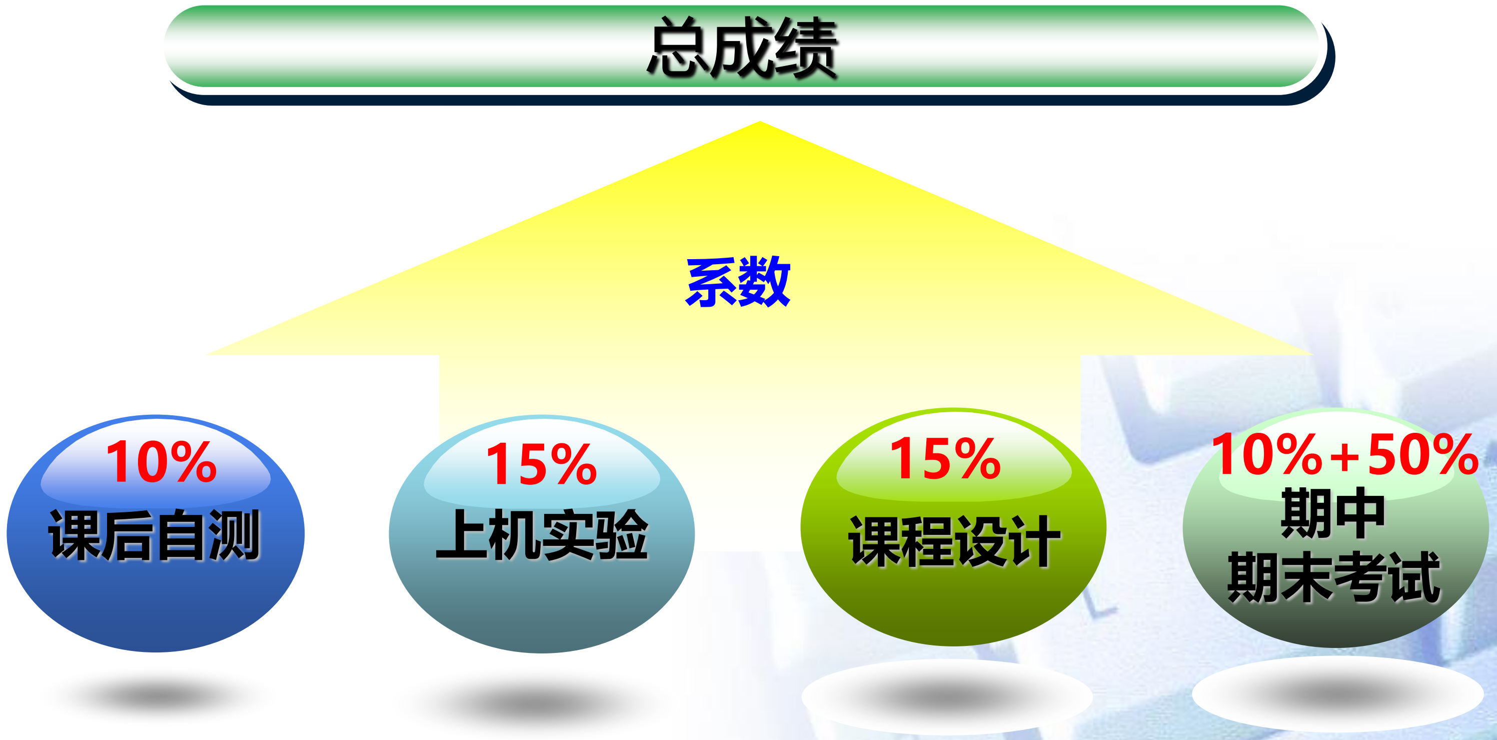
红色：考试和课堂汇报

课程安排

- **计分制**

5级11段制：A, A-, B+, B, B-, C+, C, C-, D+, D, F

- **考核划分**



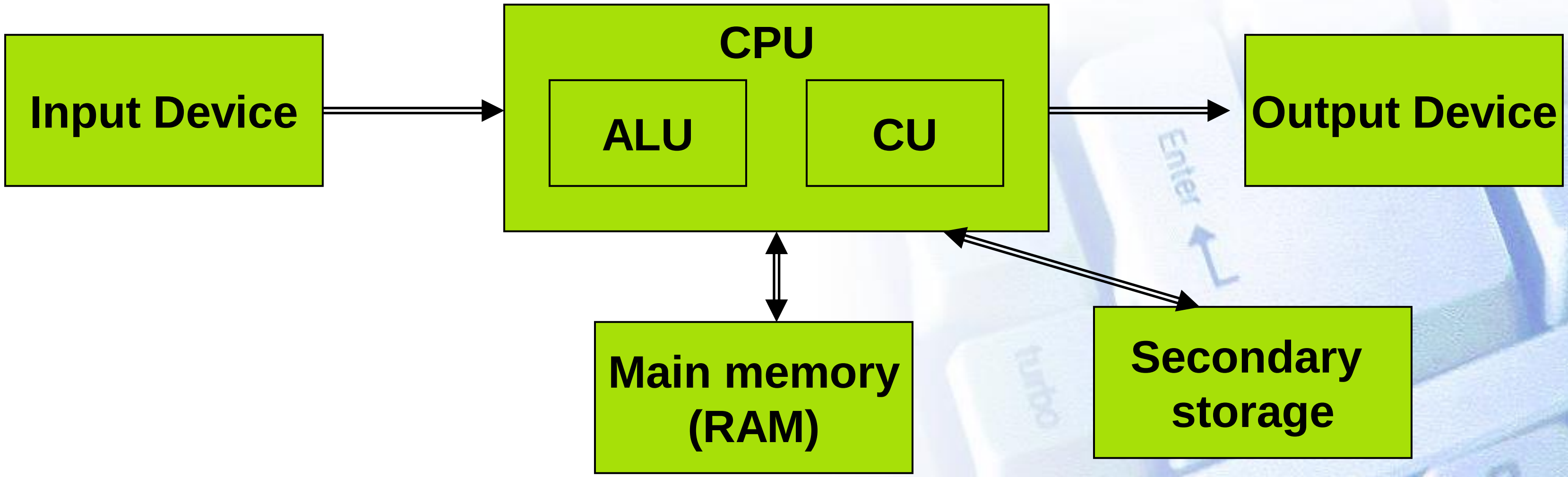
加分：参加各类程序设计竞赛获奖，酌情加分，最高不超过5分（在总成绩上直接加分）

主要内容

1. 课程介绍
2. 编程基础知识
3. C语言快速入门

冯.诺依曼体系结构

- Computing machine (Computer): "*a machine that stores and manipulates information under the control of a changeable program that is stored in its memory*."
- Pocket calculator: not a computer ! Manipulates information, but is built to do a specific task (no changeable stored program)
- This model is named the "von Neumann architecture" (冯·诺依曼 John von Neumann – 1945; EDVAC - Electronic Discrete Variable Automatic Computer – the first stored-program computer)



冯.诺依曼体系结构

- *Central Processing Unit* (CPU): the "brain" of the machine.
  - CU: Control Unit
  - ALU: Arithmetic and Logic Unit
    - Carries out all basic operations of the computer. (known as *instruction set*)
    - Examples of basic operation: adding two numbers, testing to see if two numbers are equal.
- *Main memory* (called RAM for *Random Access Memory*): stores programs and data
  - Fast but volatile
- *Secondary memory*: provides permanent storage
- Human-computer interaction: through input and output devices.
  - keyboard, mouse, monitor
  - Information from input devices is processed by the CPU and may be sent to the main or secondary memory. When information needs to be displayed, the CPU sends it to the output device(s).

计算机如何执行程序?

- How does a computer execute a program? (example programs: a computer game, a word processor, etc)
- The instructions that comprise the program are copied from the permanent secondary memory into the main memory.
- After the instructions are loaded, the CPU starts executing the program.
- For each instruction, the instruction is retrieved from memory, decoded to figure out what it represents, and the appropriate action carried out. (the *fetch- execute cycle*)
- Then the next instruction is fetched, decoded and executed.

Machine level programming

- Example: suppose we want the computer to add two numbers, and if the preliminary result is less than 10, then add 10 to the result

- The instructions that the CPU carries out might be :

```
[INSTR1]  Load into ALU the number from mem location 15
[INSTR2]  Load into ALU the number from mem location 7
[INSTR3]  Add the two numbers in the ALU
[INSTR4]  If result is bigger than 10 jump to [INSTR6]
[INSTR5]  Add 10 to the number in the ALU
[INSTR6]  Store the result from ALU into mem location 3
```

- The processors instruction set: all basic operations that can be carried out by a certain type of processor

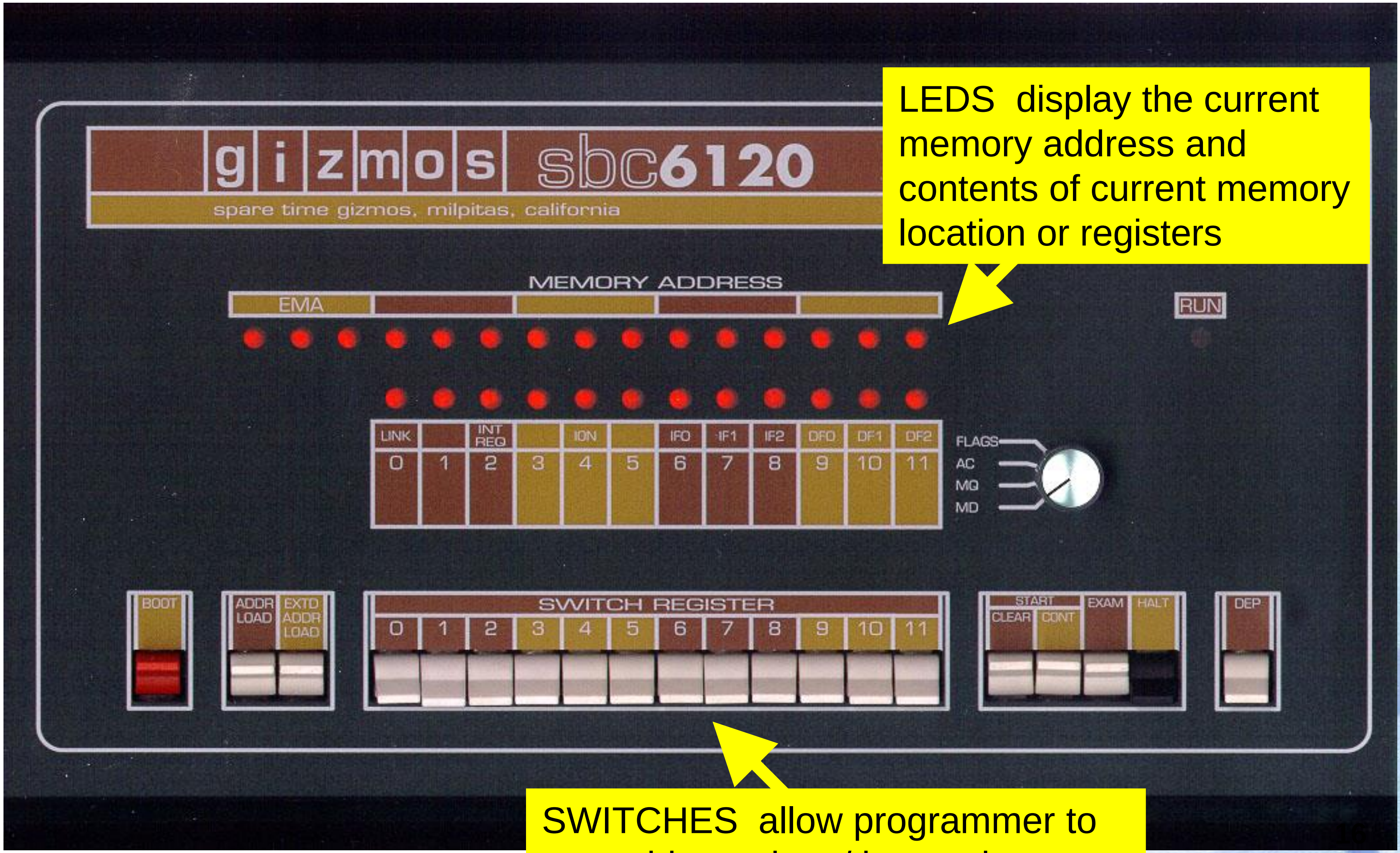
Machine level programming

- **The instructions 指令 and operands 操作数 are represented in *binary* notation (sequences of 0s and 1s) 二进制.**
  - Why binary ? Because computer hardware relies on electric/electronic circuits that have/can switch between 2 states
  - **bit** (binary digit)
  - Byte: 8 bits
- **The program carried out by the CPU, on a hypothetical processor type, could be:**

```
1010 1111
1011 0111
0111
…
```

- **This way had to be programmed in the first computers !**
- **The job of the first programmers was to code directly in machine language and to enter their programs using switches**

Example: old computer frontpanel



LEDS display the current memory address and contents of current memory location or registers

SWITCHES allow programmer to enter binary data / instructions

Higher level languages 高级语言

- **Assembly language 汇编语言**
  - First step from machine language
  - Uses ***symbolic names*** for operations
  - Example: a hypothetical assembly language program sequence:

```
1010 1111
1011 0111
0111
0011 1010
0010 1100
0110 1010
…
```

```
LD1  15
LD2   7
ADD
 CMP  10
JGE  12
 ADD 10
…
```

Higher level languages 高级语言

- **Assembly language汇编语言**
  - Translation of assembly language into machine language: in the beginning done manually, later done by a special computer program – the *assembler 汇编器*
  - Disadvantages: Low-level language:
    - programmer must learn the instruction set of the particular processor
    - Program must be rewritten in order to run on a different processor type – program is not *portable*

Higher level languages 高级语言

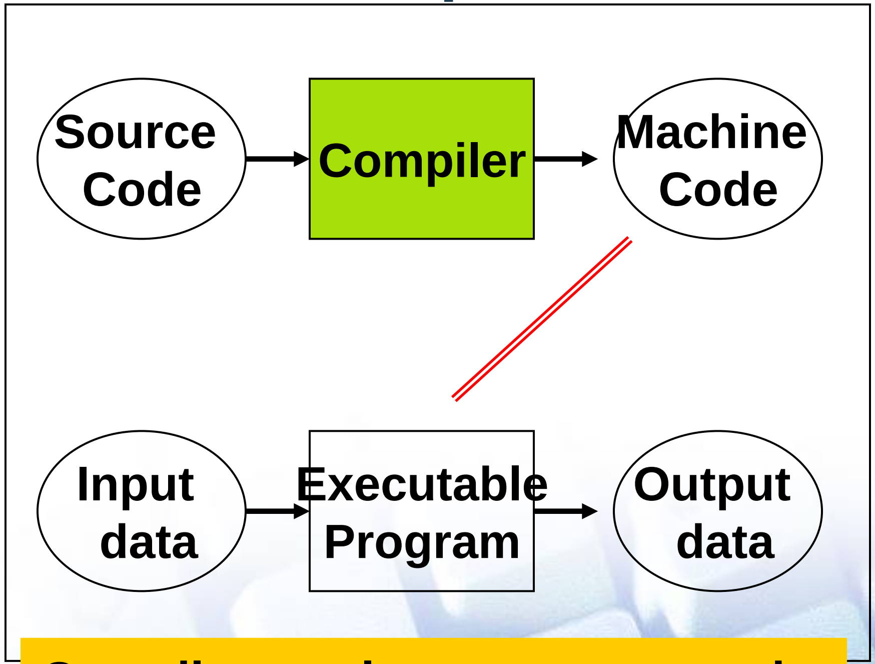
- **High level languages**
  - Using ***more abstract instructions***
  - **Portable** programs result
    - Example: a hypothetical program sequence:

```
DEFVAR a,b,c;
BEGIN
        READ a
        READ b
        READ c
        c := a+b
        IF (c <10) THEN c:=c+10
        PRINT c
END                     …
```

Higher level languages 高级语言

- **High level languages**
  - Writing portable programs, using more abstract instructions
  - A high level instruction (*statement*) is translated into many machine instructions
  - Translation of high level language into machine instructions: done by special computer programs – *compilers* 编译器

Compilers



Compiler: analyzes program and translates it into machine language
Executable program: can be run independently from compiler as many times

The C Programming Language

- Developed by Dennis Ritchie (1941-2011) at AT&T Bell Laboratories in the early 1970s
- Growth of C tightly coupled with growth of Unix: Unix was written mostly in C
- Success of PCs: need of porting C on MS-DOS
- Many providers of C compilers for many different platforms => need for standardization of the C language
- 1990: ANSI C (American National Standards Institute)
- International Standard Organization: ISO/IEC 9899:1990
- 2011: standard updated: C11, or ISO/IEC 9899:2011

主要内容

1 课程介绍

2 编程基础知识

3 C语言快速入门

C语言快速入门

第一个C语言程序

```
/*
    这个程序在标准输出（即屏幕）上打印“Hello World !”字符串
*/
#include <stdio.h>                          /* 包含文件 */
                                                   /*  空行 */
int main(void)                                 /*main 函数声明 */
{
      printf( “Hello World ! \n” );      /* 输出字符串“Hello World”  */

      return 0;                                  /* 返回一个整型值0 */
}
```

C语言快速入门

- **注释(Comment)** ：所有注释都会被预处理器拿掉，用一个空格替代
  - **/* */ 注释符**
    - 用于单行或多行注释
    - 可用于注释代码块（有风险）
    - 可用于添加解释说明
    - 注意，注释不能嵌套。第一个/*与第一个*/之间 的内容都被看做注释，不管里面有多少/*
  - **// 注释符**
    - 仅用于单行注释
    - C99及以后标准支持
  - **忠告**
    - 修改代码是注意要修改注释。不合适的注释不如没有
    - 注释的读者可能是你的Partner，后续开发者，或者几个星期后的你

```
/*
    这个程序在标准输出（即屏幕）上打印“Hello World !”字符串
*/
#include <stdio.h>                          /* 包含文件 */
                                                        /*  空行 */
int main(void)                                   /*main 函数声明 */
{
      printf( “Hello World ! \n” );      /* 输出字符串“Hello World” */

      return 0;                                     /* 返回一个整型值0 */
}
```

```
//
//  这个程序在标准输出（即屏幕）上打印“Hello World !”字符串
//
#include <stdio.h>                           // 包含文件
                                                        //  空行
int main(void)                                    // main 函数声明
{
      printf( “Hello World ! \n” );       // 输出字符串“Hello World”

      return 0;                                      // 返回一个整型值0
}
```

C语言快速入门

- 预处理指令： 被预处理器解释的指令
  - 预处理器读入源代码
  - 预处理器根据预处理指令修改源代码
  - 预处理器将修改后的源代码提交给编译器

```
/*
    这个程序在标准输出（即屏幕）上打印 “Hello World !” 字符串
*/
#include <stdio.h>                                  /* 包含文件 */
                                                                 /*  空行 */
int main(void)                                           /*main 函数声明 */
{
      printf( “Hello World ! \n” );        /* 输出字符串 “Hello World” */

      return 0;                                               /* 返回一个整型值0 */
}
```

- **main 函数**
  - 每一个C程序必须有一个main函数，是程序执行的起点
  - 语句写在函数体中（函数名后的一对花括号中），语句以分号结尾。
  - 可以一行写一个语句，也可以一行写多个语句，一个语句写在多行。
  - C语言是大小写敏感的语言，所有关键字，函数名都为小写。若将main,printf 写成大写，程序编译时将出错。

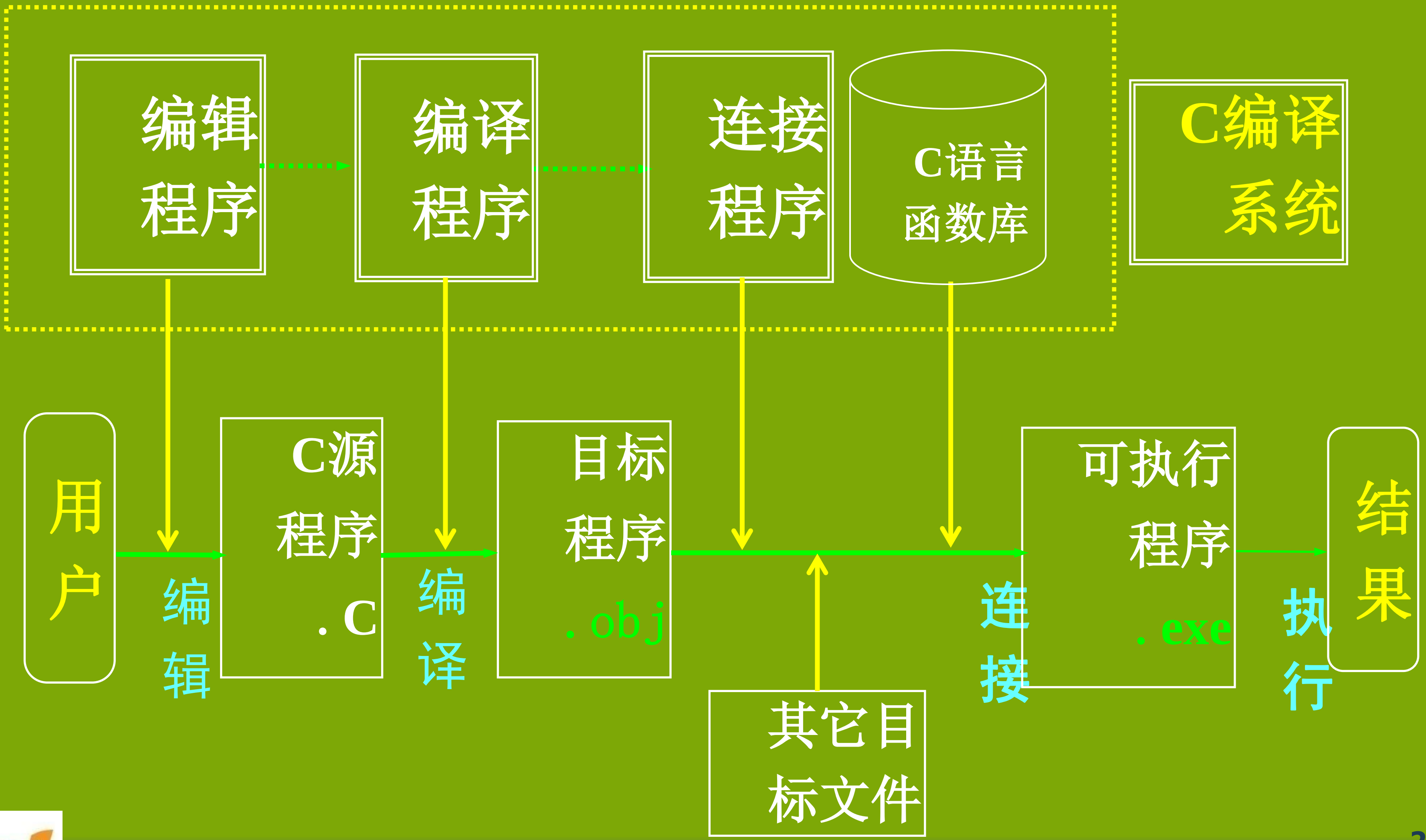
```
/*
    这个程序在标准输出（即屏幕）上打印 “Hello World !” 字符串
*/
#include <stdio.h>                                  /* 包含文件 */
                                                                 /*  空行 */
int main(void)                                           /*main 函数声明 */
{
      printf( “Hello World ! \n” );        /* 输出字符串 “Hello World” */

      return 0;                                               /* 返回一个整型值0 */
}
```

C语言开发流程

程序的实现要经过以下步骤：

- 编辑：是将写在纸上的源程序(.C)输入计算机中，并以文件的形式存放。
- 编译：经过C语言编译（编译程序）器将源程序编译成目标程序(扩展名为.OBJ)，并改正编译中出现的语法错误。
- 连接：对目标程序进行连接生成可执行文件(扩展名为.EXE)。
- 运行:运行可执行文件得到运行结果。

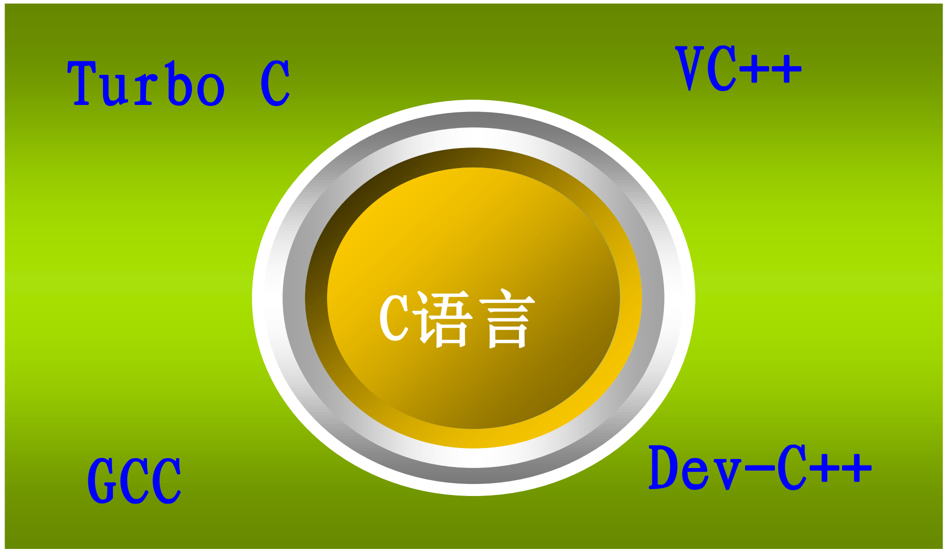
C语言开发流程



编辑程序
编译程序
连接程序
C语言函数库
C编译系统
用户
编辑
C源程序 .C
编译
目标程序 .obj
其它目标文件
连接
可执行程序 .exe
执行
结果

IDE开发环境

- IDE(Integrated Development Environment, 集成开发环境)



- 提供代码编辑器
- 提供代码编译器
- 提供调试器
- 提供友好的图形用户界面
- 集成众多的函数库
- 提供图形开发能力

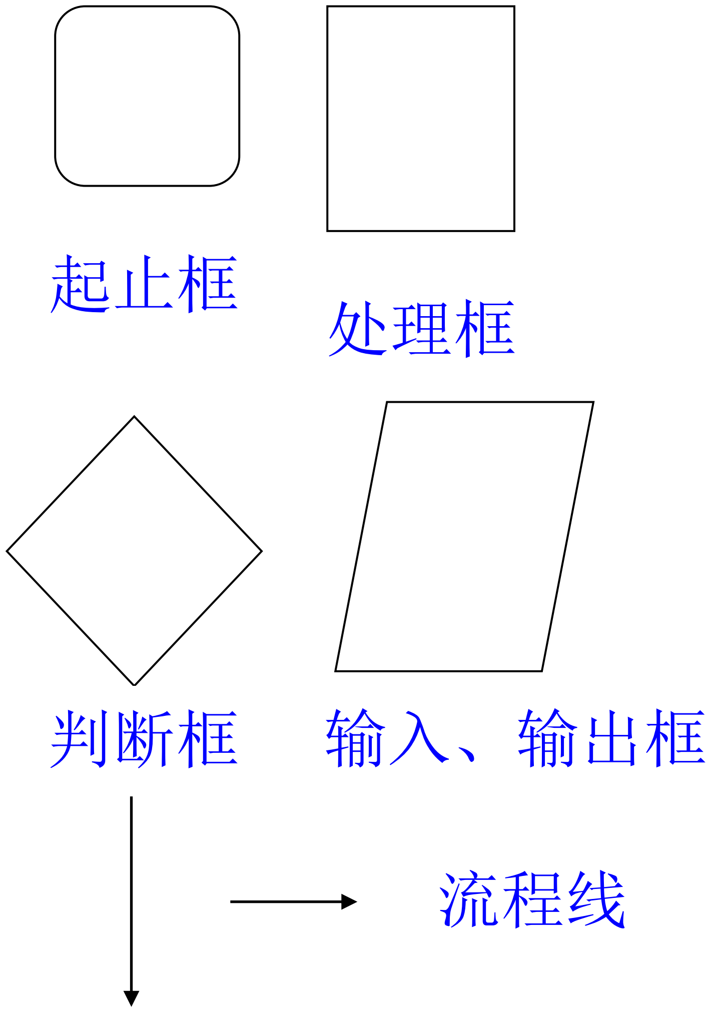
IDE 就是集成了代码**编写**功能、**分析**功能、**编译**功能、**调试**功能等一体化的**开发软件套**。

Dev-C++下载地址： https://sourceforge.net/projects/dev-cpp-2020/files/v6.5/Dev-Cpp.6.5.GCC.10.2.Setup.exe/download

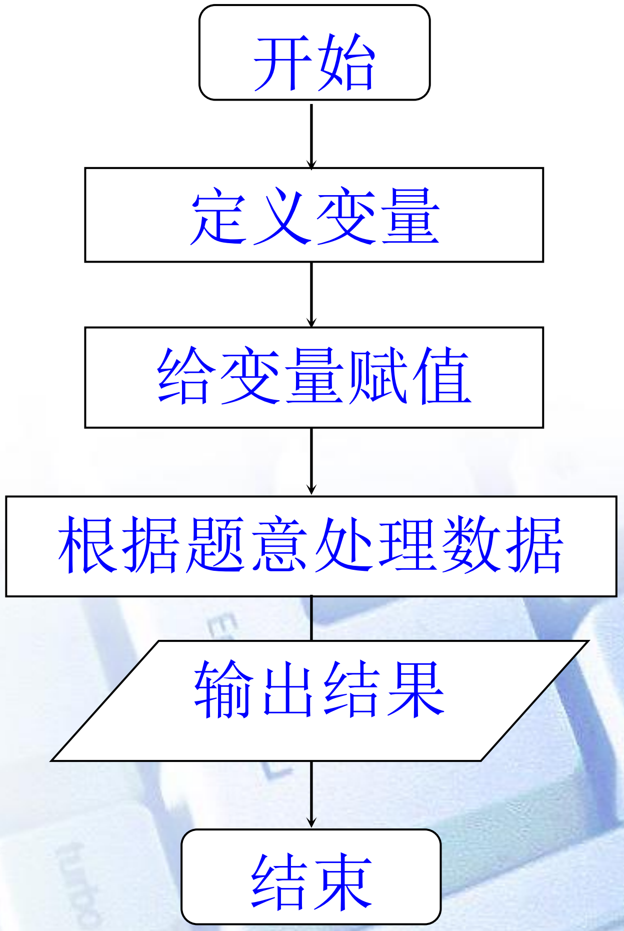
Syntax 语法 and Semantics语义

- **Syntax errors: violation of programming language rules (grammar)**
  - "*Me speak English good*."
  - Use valid C symbols in wrong places
  - Detected by the compiler
- **Semantics errors: errors in meaning:**
  - "*This sentence is excellent Italian*."
  - Programs are syntactically correct but don’t produce the expected output
  - User observes output of running program

C语言流程图



简单顺序结构的程序流程图



第二个C程序

在显示器上输出：

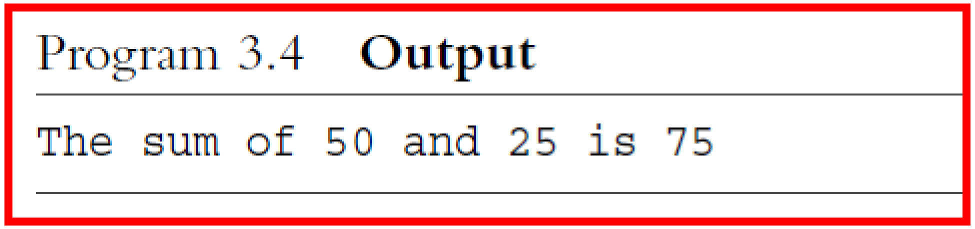
```
hello
world
```

```
#include "stdio.h"
   main()
 {
   printf("hello\n world");
}
```

该程序与第一个程序的区别只是在 hello后有个“\n”，它的功能是换行，这样 world就输出在下一行了。

定义和输出变量

```c
#include <stdio.h>
int main (void)
{
      int sum;
      sum = 50 + 25;
      printf ("The sum of 50 and 25 is %i\n", sum);
      return 0;
}
```


Program 3.4 **Output**

```
The sum of 50 and 25 is 75
```

定义int型变量sum

给变量sum赋值

使用%i来输出打印一个整型变量

输出多个变量

```
#include <stdio.h>
int main (void)
{
  int value1, value2, sum;
  value1 = 50;
  value2 = 25;
  sum = value1 + value2;
  printf ("The sum of %i and %i is %i\n",value1, value2, sum);
  return 0;
}
```

Program 3.5 **Output**

```
The sum of 50 and 25 is 75
```

格式控制符的数量与后面变量的数量必须相同

养成好的代码风格

```
main()      /* Main program starts here  */
{
   printf("Good form ");
   printf         ("can aid in ");
   printf                ("understanding a program.\n");
   printf("And bad form ");
   printf         ("can make a program ");
   printf                ("unreadable.\n");
}
```

好的代码风格

```
main( )    /* Main program starts here */{printf("Good form ");printf
   ("can aid in ");printf(" understanding a program.\n")
;printf("And bad form ");printf("can make a program ");
printf("unreadable.\n");}
```

坏的代码风格

养成好的代码风格

- **应遵循的规则**（书写清晰，便于阅读，理解，维护）
  - 一个说明或一个语句占一行
  - 用{} 括起来的部分，通常表示了程序的某一层次结构。{}一般与该结构语句的第一个字母对齐，并单独占一行。
  - 低一层次的语句或说明可比高一层次的语句或说明缩进若干格后书写。以便看起来更加清晰，增加程序的可读性。
  - 养成加注释的习惯
  - 定义变量，函数名等标示符时，尽量有意义，以便于理解

Thank you