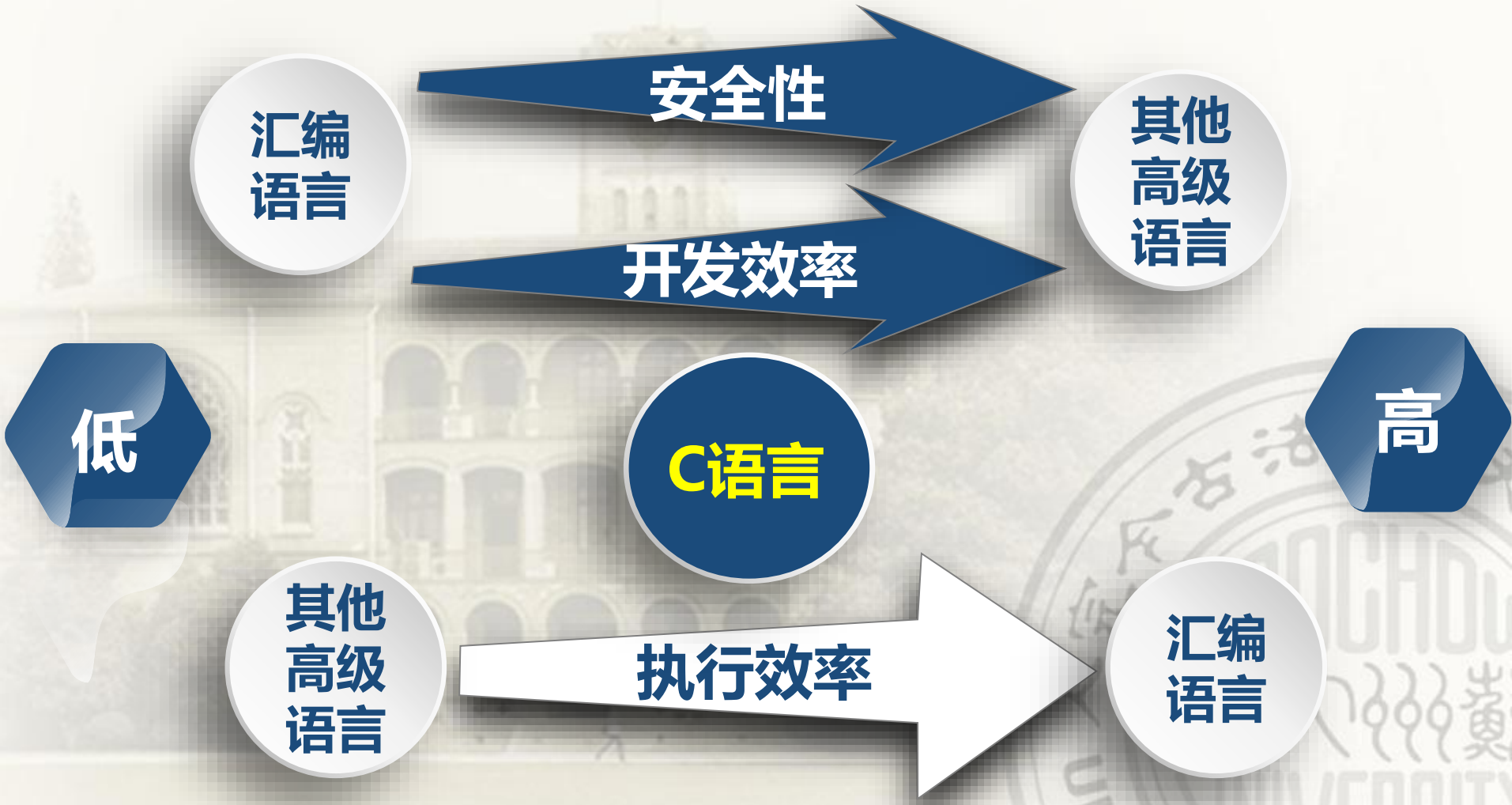
第1章 概述

1.5（补充） 学习嵌入式推荐C语言

1.5 嵌入式系统常用的C语言基本语法概要

1.5（补充）学习嵌入式推荐C语言

1.5（补充） 学习嵌入式推荐C语言——安全性、开发效率、执行效率



1.5 嵌入式系统常用的C语言基本语法概要

1.5.1 C语言运算符与数据类型——运算符

运算类型	运算符	简明含义
算术运算	+、 -、 *、 /、 %	加、减、乘、除、取模
逻辑运算	||、 &&、 !	逻辑或、逻辑与、逻辑非
关系运算	>、 <、 >=、 <=、 = =、 !=	大于、小于、大于等于、小于等于、等于、不等于
位运算	~、 <<、 >>、 &、 ^、 |	按位取反、左移、右移、按位与、按位异或、按位或
增量和减量	++、 --	增量运算符、减量运算符
复合赋值	+=、 -=、 >>=、 <<=	加法赋值、减法赋值、右移位赋值、左移位赋值
	*=、 |=、 &=、 ^=	乘法赋值、按位或赋值、按位与赋值、按位异或赋值
	%=、 /=	取模赋值、除法赋值
指针和地址	*、 &	取内容、取地址
输出格式转换	0x、0o、0b、0u	无符号十六、八、二、十进制数
	0d	带符号十进制数

1.5.1 C语言运算符与数据类型——数据类型

数据类型		简明含义	位数	字节数	值域
字节型	signed char	有符号字节型	8	1	-128～+127
	unsigned char	无符号字节型	8	1	0～255
整型	signed short	有符号短整型	16	2	-32768～+32767
	unsigned short	无符号短整型	16	2	0～65535
	signed int	有符号短整型	16	2	-32768～+32767
	unsigned int	无符号短整型	16	2	0～65535
	signed long	有符号长整型	32	4	-2147483648～+2147483647
	unsigned long	无符号长整型	32	4	0～4294967295
实型	float	浮点型	32	4	约$\pm 3.4\times(10^{-38}\sim 10^{+38})$
	double	双精度型	64	8	约$\pm 1.7\times(10^{-308}\sim 10^{+308})$

1.5.1 C语言运算符与数据类型——寄存器类型（register）变量

使用关键字“register”声明变量

使用关键字“register”声明变量后，被声明的变量采用寄存器存放，从而提高存取效率。不过由于寄存器数量有限，不能定义任意多个寄存器变量。

1.5.2 程序流程控制——顺序结构

- 顺序结构就是从前向后依次执行语句
- 从整体上看，所有程序的基本结构都是顺序结构，中间的某个过程可以是选择结构或循环结构

```
#include <stdio.h>

int main(){
    int a = 3, b = 2;
    int sum = a+b;
    printf("a+b=%d\n", sum);     //输出a+b的和
    return 0;
}
```

1.5.2 程序流程控制——选择结构

在C语言中主要有 if 和 switch 两种选择结构

```
if (表达式) 语句项;
或
if (表达式)
语句项;
else
语句项;
```

if结构

```
switch(表达式)
{
case 常数1:
    语句项1;
    break;
case常数2:
    语句项2;
    break;
  ……
default:
    语句项;
}
```

switch结构

1.5.2 程序流程控制——循环结构

C语言中的循环结构常用for循环，while循环与do...while循环

for循环：

```
for(初始化表达式；条件表达式；修正表达式)
{循环体}
```

while循环：

```
while(条件表达式)
{循环体}
```

do...while循环：

```
do
{循环体}
while(条件表达式);
```

1.5.3 函数

所谓函数把经常使用的语句群定义成函数，供其他程序调用，函数的编写与使用要遵循软件工程的基本规范。

使用函数要注意：

- 函数定义时要同时声明其类型；
- 调用函数前要先声明该函数；
- 传给函数的参数值，其类型要与函数原定义一致；
- 函数的返回值：return表达式;
- 接收函数返回值的变量，其类型也要与函数类型一致。
- 函数传参有传值与传址之分。

1.5.3 数据存储方式——总览

数组是由基本类型数据按照一定的规则组成的。

数组

指针

指针是一个用来指示一个内存地址的C语言的变量。

枚举

枚举是列出某些有穷序列集的所有成员，作为一种特定类型对象。

结构体

结构体是由一系列具有相同类型或不同类型的数据构成的数据集合。

公用体

几种不同的变量共同占用一段内存的结构，被称作“共用体”类型结构。

空类型

空类型字节长度为0，一是明确地表示一个函数不返回任何值；二是产生一个同一类型指针然后可根据需要动态地分配给其内存。

1.5.3 数据存储方式——数组

数组声明：
类型标识符 数组名[常量表达式]；（其中a为常量）

```
char c[6];
c[0] = 't' ;c[1] = 'a' ; c[2] = 'b' ;
c[3] = 'l' ; c[4] 'e' ;c[5] = '\0' ;
//字符数组c[6]中存放的就是字符串“table\0”
```

也可以通过malloc，calloc函数，进行内存空间的动态分配，从而实现数组的动态化，以满足实际需求。

```
printf("请输入所要创建的一维动态数组的长度：");
scanf("%d",&n1);
array=(int*)calloc(n1,sizeof(int)); //用calloc()函数来分配的空间，产生动态数组
```

1.5.3 数据存储方式——指针

指针的声明与赋值：

```
int a,b;          //a,b为整型数据变量
int *p1;          //声明p1是整型指针变量
p1 =&a;           //将a的地址作为p1初值
a=80;
b=*p1;          //运行结果:b=80，即为a的值
```

注意：任何类型的指针都可以直接赋值给void *类型，无需进行强制类型转换：

```
void *p1;         //声明p1无类型指针
int *p2;          //声明p2为整型指针
p1 = p2;          //用整型指针p2的值给p1直接赋值
```

1.5.3 数据存储方式——结构体

定义一个名为student的结构体变量类型：

```
struct student          //定义一个名为student的结构体变量类型
{
    char name[8];    //成员变量“name”为字符型数组
    char class[10];  //成员变量“class”为字符型数组
    int age;         //成员变量“age”为整型
};
```

实例化student的结构体类型，变量s1：

```
struct student s1;  //声明s1为“student”类型的结构体变量
```

1.5.3 数据存储方式——结构体

结构体成员的表示方式为：

结构体变量.成员名

例如，对s1的age进行访问：

```
s1.age=18;     //将数据18赋给s1.age（理解为学生s1的年龄为18）
```

结构体指针
指针访问：结构体指针名->结构体成员

```
struct  student  *Pstudent;
//声明Pstudent为一个“student”类型指针
strcpy(Pstudent->name,"LiuYuZhang");
Pstudent->age=18;
```

1.5.3 数据存储方式——枚举

enum 枚举名{
标识符[=整型常数]，标识符[=整型常数]，...
标识符[=整型常数]
} 枚举变量
注意：
枚举没有初始化，即省掉"=整型常数"时， 则从第一个标识符开始，顺次赋给标识符0, 1, 2, ...。

```
enum weekday
      { sun,mon,tue,wed,thu,fri,sat } a,b,c;//枚举类型weekday
   a=sun;      //没有赋值，默认sun=0
   b=mon;     //默认 mon=1
   c=tue ;    //默认 tue=2
```

1.5.3 数据存储方式——共用体

```
union 共用体名
{
成员表列
}变量表列;
```

```
union u                    //定义共用体u，三个数据共用一个空间。
{
    char *name;
    int age;
    int income;
}s;
  s.name= “WANGLIONG”;// 修改公用内存为“WANGLIONG”
  s.age=28;  // 又是修改了上面的公用内存,变为整型数 28
  s.income=1000; //最后修改公用内存变为 整型数1000
printf("%d\n",s.age);
                    //最后输出公共内存内容，以%d的方式就是1000了。
```

1.5.4 编译预处理——typedef定义

C语言中可以用typedef定义新的类型名，代替已有的类型名

用法说明：

（1）用typedef可以定义各种类型名，但不能用来定义变量。

（2）用typedef只是对已经存在的类型增加一个类型别名，而没有创造新的类型。

（3）typedef与#define有相似之处，但事实上它们二者不同，#define是在预编译时处理，它只能做简单的字符串替代，而typedef是在编译时处理。

（4）当不同源文件中用到各种类型数据（尤其是像数组、指针、结构体、共用体等较复杂数据类型）时，常用typedef定义一些数据类型，并把它们单独存放在一个文件中，然后在需要用到它们时，用#include命令把该文件包含进来。

（5）使用typedef有利于程序的通用与移植。。

1.5.4 编译预处理——typedef定义

```
typedef  struct  student
// 声明了新类型名STU，代表一个结构体类型。可以用该新的类型名来定义结构体变量
{
    char name[8];
    char class[10];
    int age;
}STU;
STU  student1;      //定义STU类型的结构体变量student1
STU  *S1;           //定义STU类型的结构体指针变量*S1
```

1.5.4 编译预处理——宏定义

宏定义指的是，在编译时在所有引用宏的地方，都将自动被替换成宏所代表的表达式。表达式可以是数字、字符，也可以是若干条语句。

#define 宏名 表达式

```
#define  PI  3.1415926     //以后程序中用到数字3.1415926就写PI
#define  S(r)  PI*r*r      //以后程序中用到PI*r*r就写S(r)
```

撤销宏定义：#undef 宏名

1.5.4 编译预处理——条件编译

条件编译：

```
#if  表达式      //如果表达式成立，则编译#if下的程序
#else 表达式  //否则编译#else下的程序，
#endif           //#endif为条件编译的结束标志。
```

所谓“文件包含”是指一个源文件将另一个源文件的全部内容包含进来，其一般形式为：

```
#include  “文件名”
```

谢 谢 ！