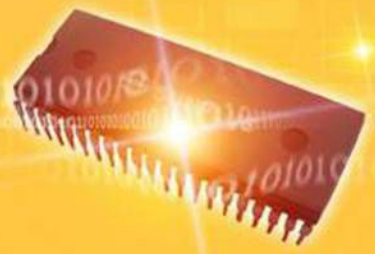
凌阳教育
www.sunplusedu.com
值得信赖的教育品牌

# 嵌入式系统工程师



Tel: 400-705-9680 , Email: edu@sunplusapp.com , BBS: bbs.sunplusedu.com

# C语言基本特性与程序架构


Tel: 400-705-9680 , Email: edu@sunplusapp.com , BBS: bbs.sunplusedu.com

# 大纲

- C语言概述
- 数据类型与混和运算
- 运算符与表达式
- C语言程序结构(顺序、选择、循环)

# C语言概述

- C语言与其它语言比较
  - **汇编语言:**

    可直接对硬件进行操作,执行效率较高;

    依赖于计算机硬件,可读性和可移植性较差.
  - **一般高级语言: C++、java、C#等**

    是在C语言的语法和基本结构上，扩展开发出来的;

    硬件操作支持性较差,运行效率不高.
- C语言同时具有汇编语言和高级语言的双重特性

  可读性好、效率高
- Linux、UNIX本身就是用C语言来编写的.

  在Linux上用C语言开发的运行效率非常高

# C语言概述

## ➢ C的关键字共有32个

- 数据类型关键字（12个）

  char, short, int, long, float, double,

  unsigned, signed, struct, union, enum, void

- 控制语句关键字（12个）

  if, else, switch, case, default

  for, do, while, break, continue, goto, return,

- 存储类关键字（5个）

  auto, extern, register, static, const

- 其他关键字（3个）

  sizeof, typedef, volatile

# C语言概述

- **结构化的语言**
- 其核心是**函数**

  一个完整的C语言程序，是由一个、且只能有一个main()函数(又称主函数，必须有)和若干个其他函数结合而成（可选）

- 函数的核心是数据+算法
  - 数据: int char float struct ...
  - 算法: 由if else switch for while等语句构成

# C语言概述

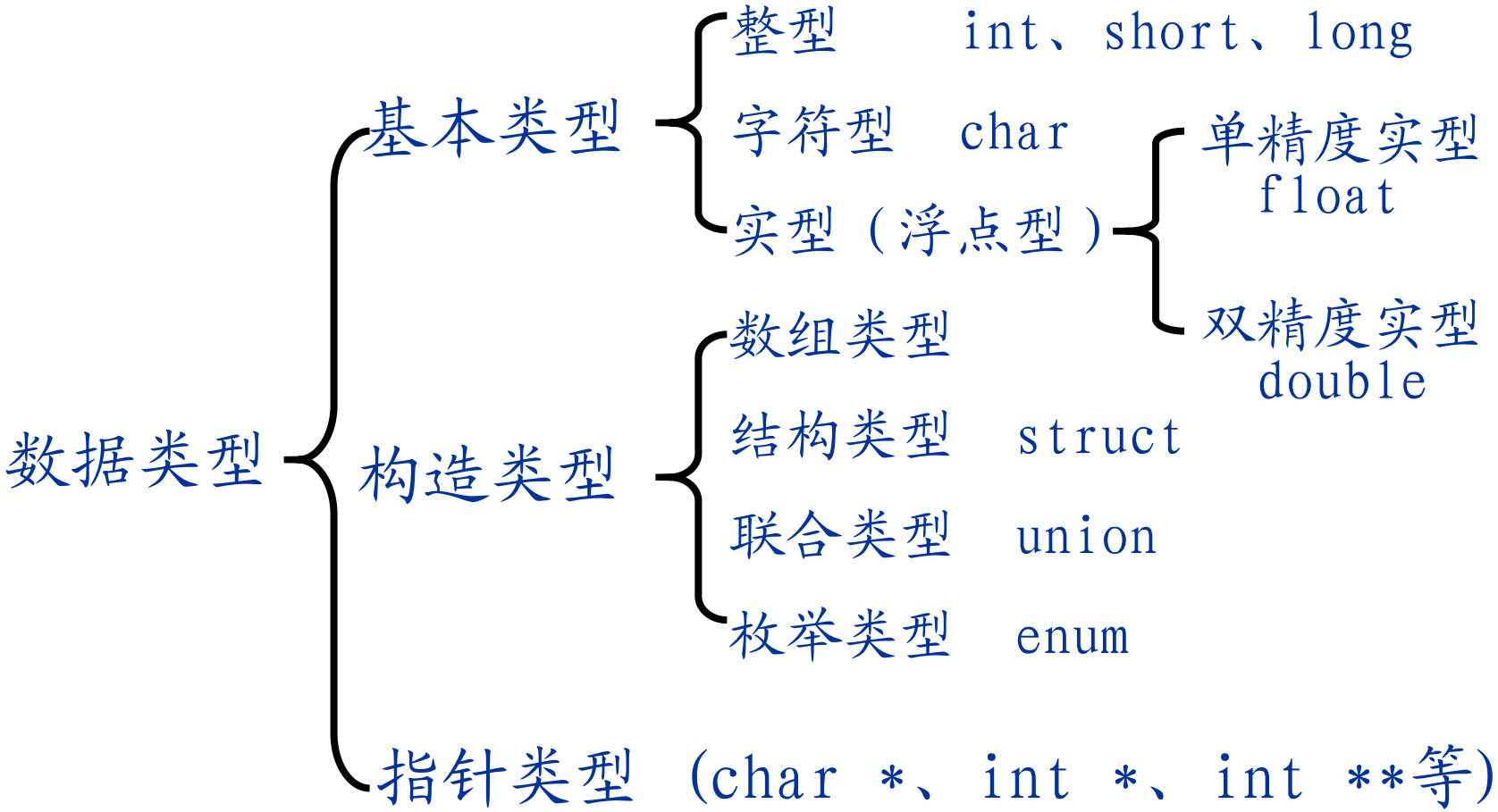
- **一个基于C的软件系统一般由以下文件构成:**
  - **若干个C文件:** 每个C文件中包含若干个函数.
  - **若干个头文件:** 每个头文件中包含一些数据结构的定义以及C函数、变量的原型声明.
  - **若干个库文件:** 库文件是编译后的二进制文件, 其中包含了若干个函数的可执行代码; 这些可执行代码在链接的时候合并到最终的可执行文件中.
  - **若干个make文件:** make文件描述了多个文件中的依赖关系以及生成最终可执行文件或库文件所需要的信息. **(windows下不可见)**
- 例子: 01_程序文件结构

# 大纲

- VC6.0使用练习
- C语言概述
- 数据类型与混和运算
- 运算符与表达式
- C语言程序结构(顺序、选择、循环)

# 数据类型

## ➢ 数据类型



# 数据类型

- 常量与变量
- 常量:

在程序运行过程中，其值不能被改变的量

- 整型　　100，125，-100，0
- 实型　　3.14 ，　0.125，-3.789
- 字符型　‘a’ ，‘b’ ，‘2’
- 字符串　“a” ，“ab” ，　“1232”
- 使用场合:

**一般出现在表达式或赋值语句中**

例: a = 100 + b;　　//100整形常量

c = 12.5;　　//实型常量

# 数据类型

- **变量:**

  在程序运行过程中，其值可以改变

- 命名规则:

  变量名只能由字母、数字、下划线组成

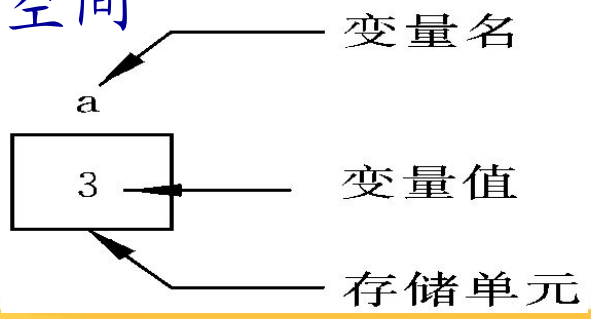
  第一个字符必须为字母或下划线;

  建议：小写英文单词+"_"线构成

- 特点:

  变量在编译时为其分配相应的内存地址

  可以通过名字和地址访问相应空间



# 数据类型

- **整型数据**
- **整型常量:（按进制分）:**

  十进制:　以正常数字1-9开头，如457 789

  八进制:　以数字0开头，如0123

  十六进制:　以0x开头，如0x1e
- **整型变量:**
  - 有/无符号短整型(un/signed) short(int)　2个字节
  - 有/无符号基本整型(un/signed) int　4个字节
  - 有/无符号长整型(un/signed) long (int)　4个字节

    (32位处理器)

# 数据类型

- **实型数据(浮点型)**
- **实型常量**
  - 实型常量也称为实数或者浮点数

    **十进制形式:** 由数字和小数点组成: 0.0、0.12、5.0

    **指数形式:** 123e3代表123*10的三次方
  - 不以f结尾的常量是double类型
  - 以f结尾的常量(如3.14f)是float类型
- **实型变量**

  单精度(float)和双精度(double)

  float型: 占4字节，7位有效数字

  double型: 占8字节，15~16位有效数字

# 数据类型

- **字符数据**
- **字符常量:**

  直接常量：用单引号括起来，如：'a'、'b'等.

  转义字符：以反斜杠“\”开头，后跟一个或几个字符、如'\n','\t'等，分别代表换行、横向跳格.

- **字符变量:**

  用char定义，每个字符变量被分配一个字节的内存空间

  字符值以ASCII码的形式存放在变量的内存单元中;

注:

```
a = 'x';
```

a变量中存放的是字符'x'的ASCII :120

即a=120跟a='x'在本质上是一致的.

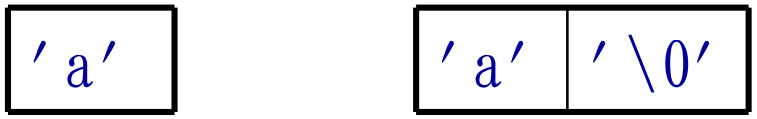
# 数据类型

## ➢ 字符串常量

是由双引号括起来的字符序列，如“CHINA”、“C program”，“$12.5”等都是合法的字符串常量.

## • 字符串常量与字符常量的不同

’a’为字符常量，“a”为字符串常量




每个字符串的结尾，编译器会自动的添加一个结束标志位'\0'，即“a”包含两个字符‘a’和’\0’

# 02.data_type.c

```c
#include <stdio.h>
int main()
{
    char a1 = '*';
    char a2[20] = "abcdef";
    unsigned short int b1 = 0x5a;
    signed int b2 = -30;
    signed long int b3 = 040;
    float c = 3.14f;
    double d = 3.1415926535898;
    double e = 314e-2;
    printf("a1=%c,%d\n",a1,sizeof(a1));
    printf("a2=%s,%d\n",a2,sizeof(a2));
    printf("b1=%x,%d\n",b1,sizeof(b1));
    printf("b2=%d,%d\n",b2,sizeof(b2));
    printf("b3=%o,%d\n",b3,sizeof(b3));
    printf("c=%.2f,%d\n",c,sizeof(c));
    printf("d=%.10f,%d\n",d,sizeof(d));
    printf("e=%.3f,%d\n",e,sizeof(e));
    return 0;
}
```

# 数据类型

- **格式化输出字符:**

| | | | |
|---|---|---|---|
| %d | 十进制有符号整数 | %u | 十进制无符号整数 |
| %x, | 以十六进制表示的整数 | %0 | 以八进制表示的整数 |
| %f | float型浮点数 | %1f | double型浮点数 |
| %e | 指数形式的浮点数 | | |
| %s | 字符串 | %c | 单个字符 |
| %p | 指针的值 | | |

- **特殊应用:**

%3d　　%03d　　%-3d　　%5.2f

# 数据类型

- **typedef 类型重定义**

  给一个已有的类型重新起个名字

  当现有类型比较复杂时，可以用typedef起个别名

  例如: unsigned short int a = 125;

  可以写为: typedef unsigned short int U16;

  U16 a=125;

- **步骤:**

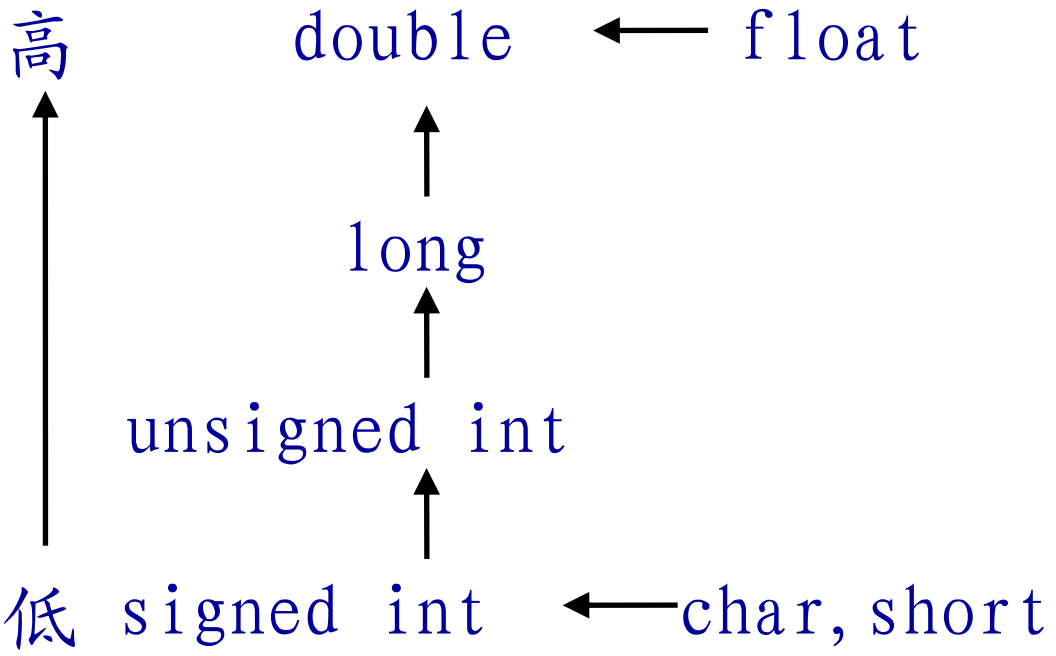
  1. 先用想要起别名的类型定义一个变量
  2. 把变量名改为我们想要起的别名（有实际意义）
  3. 在表达式最前面加一个typedef，最后面加分号
  4. 用新定义的类型名进行变量定义

# 数据的混和运算

- 数据有不同的类型，不同类型数据之间进行混合运算时必然涉及到类型的转换问题.
- 转换的方法有两种:
  - **自动转换:**
    遵循一定的规则,由编译系统自动完成.
  - **强制类型转换:**
    把表达式的运算结果强制转换成所需的数据类型.

# 数据的混和运算

- **自动转换的原则:**
  - 占用内存字节数少(值域小)的类型，向占用内存字节数多(值域大)的类型转换,以保证精度不降低.
  - 转换方向:



# 数据的混和运算

- 03.data_operation_auto.c

```
1  #include <stdio.h>
2  int main()
3  {
4      int num=5;
5      printf("s1=%d\n",num/2);
6      printf("s2=%lf\n",num/2.0);
7      return 0;
8  }
```

- num为整型,与2相除仍为整型，与2.0相除结果为double型

# 数据的混和运算

- **强制转换:** 通过类型转换运算来实现

  (类型说明符) (表达式)

  功能:

  把表达式的运算结果强制转换成类型说明符所表示的类型

  例如:

```
(float)a;      // 把a的值转换为实型
(int)(x+y);    // 把x+y的结果值转换为整型
```

- 注意:

  类型说明符必须加括号

# 数据的混和运算

➢ 04.data_operation_force.c

```
1  #include <stdio.h>
2  int main()
3  {
4      float x=0;
5      int i = 0;
6      x=3.6f;
7      i = x;              //x为实型,i为整型，直接赋值会有警告
8      i = (int)x;         //使用强制类型转换
9      printf("x=%f, i=%d\n",x,i);
10     return 0;
11 }
```

# 数据的混和运算

无论是强制转换或是自动转换，都只是为了本次运算的需要，而对变量的数据长度进行的临时性转换，而不改变数据定义的类型.

# 大纲

- VC6.0使用练习
- C语言概述
- 数据类型与混和运算
- 运算符与表达式
- C语言程序结构(顺序、选择、循环)

# 运算符与表达式

- **运算符**

  用算术运算符将运算对象(也称操作数）连接起来的、符合C语法规则的式子，称为C算术表达式

- 运算对象包括常量、变量、函数等

  **例如：** a * b / c - 1.5 + 'a'

# 运算符与表达式

- C语言常用运算符
  1. 算术运算符 （+、-、*、/、%）
  2. 关系运算符 （>、<、==、>=、<=、!= ）
  3. 逻辑运算符 （！、&&、||）
  4. 位运算符 （<<、>>、&、|、~ 、∧ ）
  5. 赋值运算符 （=及其扩展赋值运算符）
  6. 条件运算符 （？：）
  7. 逗号运算符 （，）

# 运算符与表达式

8. 指针运算符 （*和&）
9. 求字节数运算符（sizeof(类型或变量））
10. 强制类型转换运算符（(变量或常量)）
11. 分量运算符（. ->）
12. 下标运算符（ [ ] ）
13. 其他（ 如函数调用运算符（ ））

# 运算符与表达式

## ➢ 复合赋值运算符

在赋值符“=”之前加上其它二目运算符可构成复合赋值符：+=, -=, *=, %=, <<=, >>=, &=, ^=

```
a  +=  5          //等价于 a = a+5;
x  *=  y+7        //等价于x = x*(y+7)
```

# 运算符与表达式

## ➢ 自增、自减运算符

作用是使**变量**的值增 1 或减 1

- ++i, --i（先加/减，后使用）
- i++, i--（先使用，后加/减）

➢ 例如: i = 3;

① j=++i;

i的值先变成4，再赋给 j；此时i, j的值均为 4 .

② j=i++;

先将 i的值3赋给 j，j 的值为 3，然后 i 变为 4 .

# 运算符与表达式

## ➢ 位运算

位运算是指按二进制位进行的运算

C语言不直接支持位运算，必须借助于位运算完成

➢ 例如:

将一个存储单元中的各二进制位左移或右移一位

将一个存储单元的指定位清零或置位(写1)

➢ C语言提供的位运算符:

| 运算符 | 含义 | 运算符 | 含义 |
|---|---|---|---|
| & | 按位与 | ~ | 取反 |
| \| | 按位或 | << | 左移 |
| ^ | 按位异或 | >> | 右移 |

# 运算符与表达式

- 参加运算的两个数据，按二进制位展开，然后进行相应的运算，如：“与、或、异或、非”等
  - & 常用来将指定位清零

    ```
    a = a & 0xf0;
    ```
  - | 常用来将指定位置1

    ```
    a = a | 0xf0;
    ```
  - ^ 判断两个位值，不同为1，相同为0，常用来使特定位翻转等

    ```
    a = a ^ 0xf0;
    ```
  - ~ 按位取反，即将0变1或1变0，一般配合其它位运算符使用，以达到屏蔽某些关键位

    ```
    a = ~ a;
    ```

# 运算符与表达式

- **移位运算**
  - **左移<<**

    将一个数的二进制位左移，高位丢弃，低位补0

    例：a = a<<2　将a的二进制数左移2位，右补0

    若a =15，即二进制数0 0 0 0 1 1 1 1

    左移2位得0 0 1 1 1 1 0 0，（十进制数60）
  - **>>算术右移**

    有符号数,如果为正数,则左边移入0，右边丢弃

    如果为负数，左边移入1，右边丢弃
  - **>>逻辑右移**
    - 不关心正数、负数，左边均移入0，右边丢弃

**算术或者逻辑右移，由编译器决定**

# 运算符与表达式

05.bit.c

```c
#include <stdio.h>
int main()
{    //0101 0101
     unsigned char a1=0x55,a2=0x55,a3=0x55,a4=0x55;
     unsigned char a5=0x55,a6=0x55,a7=0x55;
     printf("a1=%0x\n",a1);
     a2 = a2 & 0xf0;
     printf("a2=%0x\n",a2);
     a3 = a3 | 0x0f;
     printf("a3=%0x\n",a3);
     a4 = a4 ^ 0x0f;
     printf("a4=%0x\n",a4);
     a5 = ~a5;
     printf("a5=%0x\n",a5);
     a6 = a6 >> 2;
     printf("a6=%0x\n",a6);
     a7 = a7 <<2;
     printf("a7=%0x\n",a7);
     return 0;
}
```

# 运算符与表达式

- **运算符的优先级:**
  - C语言中，运算符的运算优先级共分为15级. 1级最高，15级最低.
  - 优先级较高的先于优先级较低的进行运算.
  - 在一个运算量两侧的运算符优先级相同时，则按运算符的结合性所规定的结合方向处理.
- **各运算符的结合性:**

  左结合性(从左至右)： + - * /

  右结合性(从右至左)： = ++ --

# 运算符与表达式

- 在判断同优先级运算符计算顺序时，要注意结合性，详细的优先级及结合性请参考以下表格：

| 优先级别 | 运算符 | 运算形式 | 结合方向 | 名称或含义 |
|---|---|---|---|---|
| 1 | () | (e) | 自左至右 | 圆括号 |
| | [] | a[e] | | 数组下标 |
| | . | x.y | | 成员运算符 |
| | -> | p->x | | 用指针访问成员的指向运算符 |
| 2 | - + | -e | 自右至左 | 负号和正号 |
| | ++ -- | ++x 或 x++ | | 自增运算和自减运算 |
| | ! | ! e | | 逻辑非 |
| | ~ | ~e | | 按位取反 |
| | (t) | (t)e | | 类型转换 |
| | * | * p | | 指针运算，由地址求内容 |
| | & | &x | | 求变量的地址 |
| | sizeof | sizeof(t) | | 求某类型变量的长度 |

# 运算符与表达式

| | | | | |
|---|---|---|---|---|
| 3 | * / % | e1 * e2 | 自左至右 | 乘、除和求余 |
| 4 | + - | e1+e2 | 自左至右 | 加和减 |
| 5 | << >> | e1<<e2 | 自左至右 | 左移和右移 |
| 6 | < <= > >= | e1<e2 | 自左至右 | 关系运算(比较) |
| 7 | == != | e1==e2 | 自左至右 | 等于和不等于比较 |
| 8 | & | e1&e2 | 自左至右 | 按位与 |
| 9 | ^ | e1^e2 | 自左至右 | 按位异或 |
| 10 | \| | e1\|e2 | 自左至右 | 按位或 |
| 11 | && | e1 && e2 | 自左至右 | 逻辑与(并且) |
| 12 | \|\| | e1\|\|e2 | 自左至右 | 逻辑或(或者) |
| 13 | ? : | e1? e2: e3 | 自右至左 | 条件运算 |
| 14 | =<br>+= -= *=<br>/= %= >>=<br><<= &= ^=<br>\|= | x=e<br>x+=e | 自右至左 | 赋值运算<br>复合赋值运算 |
| 15 | , | e1, e2 | 自左至右 | 顺序求值运算 |

# 运算符与表达式

- 实际使用时，我们无需过多考虑优先级的问题
- 在我们优先运算的表达式表达式之间加入()即可
- **例:**

```
e = a + b - c * d; //先算乘后加、减

e =(a +(b - c))* d; //先减，再加，最后乘
```

# 大纲

- VC6.0使用练习
- C语言概述
- 数据类型与混和运算
- 运算符与表达式
- C语言程序结构(顺序、选择、循环)

# C语言程序结构

- C语言支持最基本的三种程序运行结构

  顺序结构、选择结构、循环结构

  - **顺序结构:**

    程序按顺序执行，不发生跳转.

  - **选择结构:**

    依据是否满足条件，有选择的执行相应功能.

  - **循环结构:**

    依据条件是否满足，循环多次执行某段代码.

# C语言程序结构

## ➢ 选择结构:

依据是否满足条件，有选择的执行相应功能.

① if（表达式）语句;

② if（表达式）语句1 else 语句2;

③ if（表达式1）语句1;
else if（表达式2）语句2;
else if（表达式3）语句3;
...
else 语句n;

① 条件运算符 （a>b）？a:b;

说明：条件为真，表达式取值a，否则取值b

# C语言程序结构

```
switch（表达式）
{
    case 常量表达式1:
            语句1;
            break;
    case 常量表达式2:
            语句2;
            break;
   default: 语句3; break;
}
```

注意：break的使用

# C语言程序结构

- 练习1:

  从屏幕上输入一个学生的成绩(0-100)

  对学生成绩进行评定:

  <=60为"E"

  60~69为"D"

  70~79为"C"

  80~89为"B"

  90以上为"A"

  <0或>100提示成绩输入出错

使用: if else if等实现

# C语言程序结构

- 练习2:

  从键盘输入1¯7的数字，分别提示

  Monday、Tuesday、Wednesday、Thursday、friday、saturday、sunday

  输入其它，提示出错

  使用: switch case break 实现

# C语言程序结构

- **循环结构**
  - ① for ( ; ; ) { }
  - ② while语句 先判断表达式后执行.
  - ③ do-while语句 先执行语句后判断表达式.
  - ④ goto语句 实现程序跳转（尽量少使用）.

- **break语句和continue语句的区别**

  break语句用于跳出本层循环.

  continue用于结束本次循环.

# C语言程序结构

- 练习3:

  输出100以内能被7整除的数

  分别用for循环和while循环完成

- 练习4:

  要求练习2可以循环输入

  当输入0时退出循环: break

  输入0-7以外的数字提示重新输入: continue

凌阳教育
www.sunplusedu.com

值得信赖的教育品牌

Tel: 400-705-9680 , Email: edu@sunplusapp.com , BBS: bbs.sunplusedu.com