第三章 数据类型和变量

一、变量的定义

二、基本数据类型

三、变量命名规则、变量初始化

四、标准输入输出

五、枚举类型

六、数组

一、变量的定义

1.1 为什么引入变量?

在程序开发中，经常需要频繁的记录和使用某一个值，通常的做法不是直接使用这个值，而是定义一个变量来存储它的值，这样使得我们更容易去引用和修改这个值。

例如要记录一个数值5，我们可以进行如下定义:

```
int a = 5;
```

说明:

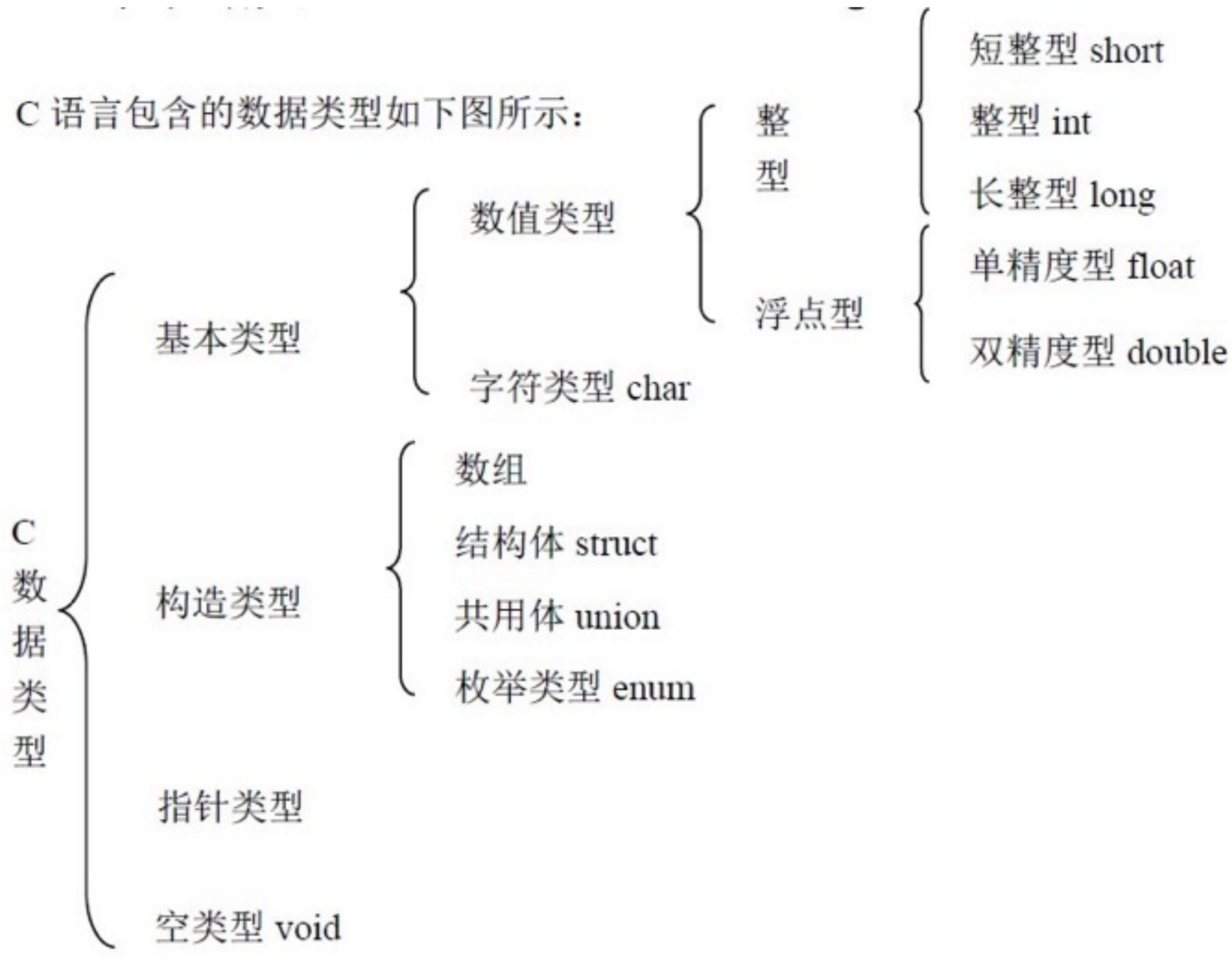
- 创建一个变量叫做a，用于存储一个整型数值;
- 这个变量是由变量名（a）和类型（int）构成;
- 该变量存储了一个数值为5;
- 可以通过变量名a来访问该变量存储的值，如: `printf("%d",a);`

注意：变量的本质是内存中的存储单元，任意一个变量在内存中都需要占用一定的内存空间，并且不同类型的数据变量在内存中占用的字节数不一样。

二、基本数据类型

2.1 C语言的数据类型分类

C 语言包含的数据类型如下图所示：



2.2 基本数据类型在内存中占用的字节数

Type（数据类型）	Size（32位系统）	Size（64位系统）
char,unsigned char字符	1 byte	1 byte
short,unsigned short短整型	2 bytes	2 bytes
int,unsigned int无符号整数	4 bytes	4 bytes
long, unsigned long长整型	4 bytes	8 bytes
float 单精度浮点数	4 bytes	4 bytes
double 双精度浮点数	8 bytes	8 bytes

2.3 sizeof操作符

sizeof是C/C++中的一个操作符（operator），简单的说其作用就是返回数据类型在内存中占用的字节数。

```
#include <stdio.h>
int main(int argc, const char * argv[]) {
    char c = 'A';//字符型
    short s = 5; //短整型
    int a = 10;  //整型
    long l = 20; //长整型
    float pi = 3.14;//单精度浮点型
    double d = 3.14159;//双精度浮点型
    printf("%d,%d,%d,%d,%d,%d",sizeof(c),sizeof(s),
         sizeof(a),sizeof(l),sizeof(pi),sizeof(d));
    return 0;
}
```

程序运行结果如下：

1,2,4,4,4,8（32位）

1,2,4,8,4,8（64位）

三、变量命名规则

3.1 命名规则

- 每个变量必须以字母或下划线作为开头，可由字母、下划线、数字组成；
- C语言区分大小写，如 int A,int a是两个不同的变量；
- 不能以C语言的关键字作为变量名，如 main、while、switch、case、if等；

同时变量的命名还应遵守以下原则：

⑴ 命名应当直观且可以拼读，可望文知意，便于记忆和阅读。

⑵ 命名的长度应当符合“min-length && max-information”原则。C 是一种简洁的语言，命名也应该是简洁的。例如变量名MaxVal 就比 MaxValueUntilOverflow 好用。标识符的长度一般不要过长。

⑶ 当标识符由多个词组成时，除开头外的每个单词的第一个字母应该大写，其余全部小写。比如：int currentVal;

3.2 变量的初始化

变量的初始化即给变量赋最初的值，变量的初始化有三种方式：

- 方式一：声明的同时进行初始化。

  如： `int x = 5; char code = 'B';`

- 方式二：先声明，再通过赋值语句赋值。

  如： `int x ; x = 5;`

- 方式三：声明后通过用户输入值进行初始化。

  如： `int x ; scanf("%d",&x);`

注意：在C语言等高级语言中，为每一个变量赋初值被视为良好的编程习惯，有助于减少出现Bug的可能性。

3.3 常量 constants

在C语言可以使用const关键字来定义常量，常量意味着初始化完成后，其值不能再被修改，因此常量必须要求在声明时进行初始化。

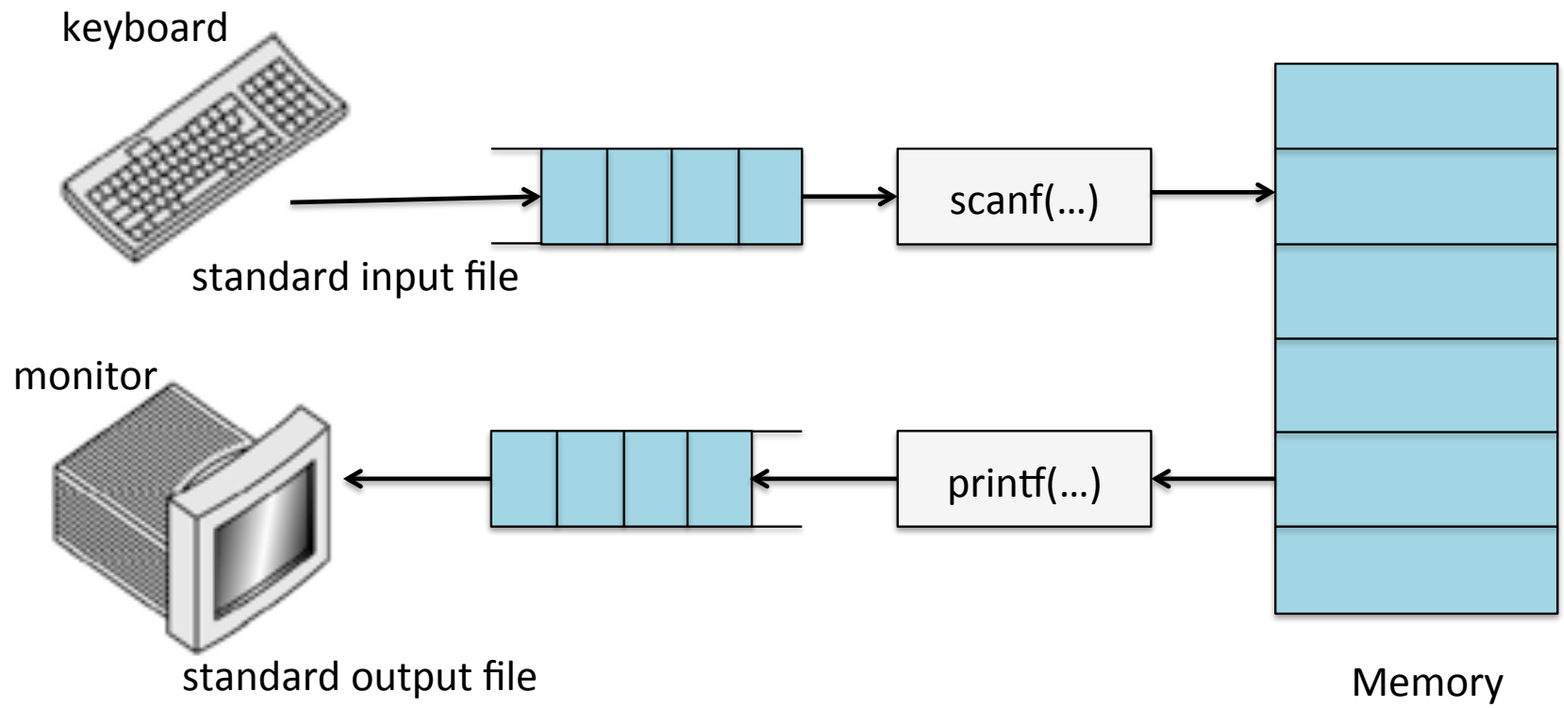
```c
#include <stdio.h>

int main(int argc, const char * argv[]) {

    const float pi = 3.14;
    pi = 3.14159; //错误 Read-only variable is not assignable
    return 0;
}
```

Q：什么情况下程序中会使用到常量?

当我们编写程序时，不期望在后面的代码或程序执行的过程中修改某一个变量的值时，可以使用“常量”以增加程序的健壮性。

四、标准输入输出

`#include <stdio.h>` stdio 就是指“standard input & output”（标准输入输出），所以源代码中如用到标准输入输出函数时，就要包含这个头文件。



标准输入输出图解

4.1 标准输入输出示例：

以下程序代码演示了一个完整的标准输入输出的过程：

```
#include <stdio.h>//包含标准输入输出头文件

int main(int argc, const char * argv[]) {
    int a;//编译器给变量a在内存中分配了4个字节的内存空间,但是它的值是未知
的
    scanf("%d",&a);//等待用户从键盘输入一个值,存储到内存中
    printf("a=%d\n",a);//将变量的值从内存中取出来打印输出在屏幕中

    return 0;
}
```

4.2 standard output file

标准输出：

```
#include <stdio.h>
printf(format string,data list);
```

printf函数的两种使用：

1、直接打印一串字符 printf(“…”)，如：printf(“Hello,world!\n”);

2、打印变量的值，如:printf(“%d,%d”, a,b);

```
#include <stdio.h>

int main(int argc, const char * argv[]) {
    int a = 5;
    float pi = 3.14;
    printf("下面将打印两个变量的值: \n");//直接打印输出一个字符串
    printf("%d,%f\n",a,pi);//打印两个变量的值
    return 0;
}
```

程序运行结果如下：

下面将打印两个变量的值:
5,3.140000

4.3 输出格式控制符

C语言中使用printf打印某一类型的变量的值时，printf函数的第一个参数“”中必须使用该变量对应的格式控制符,如打印一个整型变量则使用%d，printf(“%d”,a);

Type（数据类型）	fromat（格式控制符）
char	%c
short	%hd
int	%d 或 %i
long	%ld
float	%f
double	%f 或 %lf

4.4 standard input file

标准输入：与printf函数一样，都被定义在头文件stdio.h里，因此在使用scanf函数时要加上#include <stdio.h>。它是格式输入函数，即按用户指定的格式从键盘上把数据输入到指定的变量之中。

```
#include <stdio.h>
scanf(format string,address list);   //第二个参数必须是变量的地址
```

```
#include <stdio.h>
int main(int argc, const char * argv[]) {
    int a; float pi;
    printf("请从键盘输入两个变量的值：\n");//直接打印输出一个字符串
    scanf("%d%f",&a,&pi);//等待用户从键盘输入值
    printf("%d,%f\n",a,pi);//打印两个变量的值
    return 0;
}
```

注意：使用scanf函数时,双引号中尽量只写变量对应的格式控制符,不要添加其它额外的内容，否则用户必须按照双引号中填写的格式进行输入。

如 scanf(“a=%d,b=%d”,&a,&b),则用户必须在键盘中键入a=5,b=10的格式。

4.5 输入格式控制符

scanf函数中变量对应的格式控制符与printf几乎一样，但除了double类型，double类型对应的格式控制符为 %lf ，如 double d; scanf(“%lf”,&d);

Type（数据类型）	fromat（格式控制符）
char	%c
short	%hd
int	%d 或 %i
long	%ld
float	%f
double	%lf

五、枚举类型

5.1 枚举类型的概念

Q：为什么要使用枚举类型?

在实际编码中,有些变量的取值被限定在一个有限的范围内。例如,一个星期内只有七天,一年只有十二个月,一个班每周有六门课程等等。如果把这些量定义为整型,字符型或其它类型显然是不妥当的。为此,C语言提供了一种称为“枚举”的类型,它可以限定变量的取值范围，**但其主要作用在于增加程序的可读性。**

在“枚举”类型的定义中要列举出它的所有可能的取值,限定该“枚举”类型的变量取值不能超过其定义的范围。

枚举的使用其实是定义一些标签（通常是一个字符或者字符串）来代替整型值，在本质上，枚举类型的取值范围是整型的一个子集。

5.2 定义枚举类型

在程序开发中，如果使用整型来表示人的性别，则可能如下定义：

```
int sex;
sex = 0;//表示女性
sex = 1;//表示男性
```

这显然不能很好的去管理这个变量，首先程序员必须记住0和1所代表的含义，其次在编写或执行代码的过程中可能会有意或无意中改变了sex为其它的值，造成程序出错。因此C语言中提供的枚举类型就可以很好的解决这个问题：

```
#include <stdio.h>

int main(int argc, const char * argv[]) {
    //定义一个枚举类型sex，该类型的变量的取值是0、1、2，即girl的默认值为0，以后依次递增
    enum sex {girl,boy,unknown};
    enum sex zhang = girl;
    printf("%d\n",zhang);//输出结果为0
    return 0;
}
```

5.3 声明枚举类型的二种方式:

方式一: 先声明枚举类型, 再定义变量

```
enum sex {girl,boy,unknown};//声明枚举类型sex
enum sex zhang = girl;//声明枚举变量zhange并赋值为girl
```

方式二: 声明枚举类型的同时声明枚举变量

```
enum sex {girl,boy,unknown}zhang;//声明类型和变量
zhang = girl;//给枚举变量赋值
```

对方式二中的声明方式, 可在声明枚举类型时省略类型名称sex,

```
enum {girl,boy,unknown}zhang;//省略类型名称
```

课堂练习:

定义一个枚举类型来表示以下几种颜色, 红黄蓝白绿。

5.3 枚举类型的取值

枚举类型中标签的默认值从0开始，后面依次递增1，即枚举sex的取值是0、1、2，girl等于0； `enum sex {girl,boy,unknown};`

当然枚举类型中标签的值也可以被显式地修改，如：

```
enum sex {girl=10,boy,unknown};//boy的值变为11
```

示例代码：

```
#include <stdio.h>

int main(int argc, const char * argv[]) {
    enum sex {girl=10,boy,unknown};
    enum sex zhang = boy;
    printf("zhange = %d",zhang);
    return 0;
}
```

程序运行结果如下：

```
zhang = 11
```

六、数组类型

6.1 数组的概念

什么是数组？ 数组是相同类型的值的集合。在程序设计中,为了处理方便,把具有相同类型的若干变量按有序的形式组织起来。这些按序排列的同类型数据元素的集合称为数组。

例如:你可能想要创建一个具有5个整数的集合。

有一种方式可以完成:可以直接声明5个整型变量,把他们放在一起使用。

`int a, b, c, d, e;` 这样做是可以的。但是如果你想要创建1000个整数的集合呢? 一种更简单的方式完成—去创建一个具有1000个整型数值的数组变量即可。

例如: `int a[1000];`

6.2 声明数组变量

```
int a[5];
```

说明：

1、int a[5]表示声明一个数组变量，存储5个整型元素；

2、a表示数组变量名；

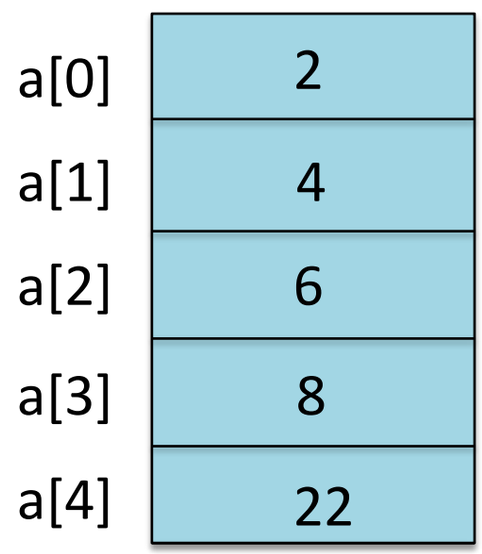
3、[5]表示元素的个数为5；

6.3 数组变量初始化

```
int a[5] = {2,4,6,8,22};
```

数组中的元素访问形式： 数组[下标索引]

下标从0开始到n-1



6.4 二维数组

二维数组本质上是以数组作为数组元素的数组，即二维数组是由多个一维数组构成的“数组的数组”。

```
int a[2][3];
```

说明：

1、int a[2][3]表示声明一个二维数组变量，存储2*3 = 6 个整型元素；

2、可以把该二维数组理解为有二行三列的元素或有两个“元素个数为3”的一维数组构成

3、数组的第一个元素为a[0][0]，第二个为a[0][1]，第三个为a[0][2]，第四个为a[1][0]，以次类推。

行/列	第一列	第二列	第三列
第一行a[0]	a[0][0]	a[0][1]	a[0][2]
第二行a[1]	a[1][0]	a[1][1]	a[1][2]

6.5 二维数组初始化

二维数组初始化也是在类型说明时给各下标变量赋以初值。二维数组可按行分段赋值，也可按行连续赋值。

- 按行连续赋值可写为：

```
int a[2][3] = {2,4,6,8,22,30};
```

行/列	第一列	第二列	第三列
第一行a[0]	2	4	6
第二行a[1]	8	22	30

- 按行分段赋值可写为：

```
int a[2][3] = {{2,4,6},{8,22,30}};
```

数组是一种构造类型的数据。二维数组可以看作是由一维数组的嵌套而构成的。设一维数组的每个元素都又是一个数组，就组成了二维数组。当然，前提是各元素类型必须相同。根据这样的分析，一个二维数组也可以分解为多个一维数组。C语言允许这种分解。

The End

渥瑞达·教学部