

第 3 讲

数据组织与处理基础

主要内容

- 3.1 数据类型的概念
- 3.2 一般数据的组织与处理
- 3.3 批量数据的组织与处理
- 3.4 混合数据的组织与处理
- 3.5 动态数据的组织与处理
- 3.6 磁盘数据的组织与处理

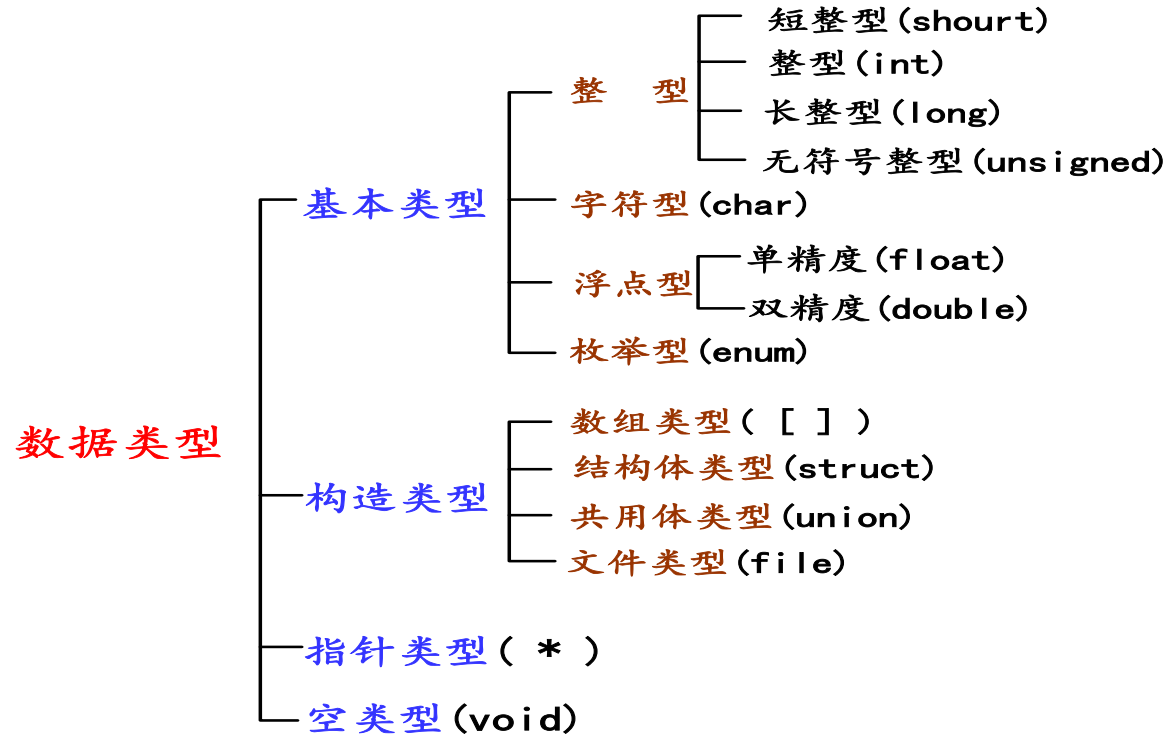
3.1 数据类型的概念

In God we trust, everyone else must bring data.

- 不同数据间的相互运算，在机器内部的执行方式是不一样的。这就要用户先定义数据的特性再进行操作，这里的特性也就是数据类型。数据类型是编程语言中为对数据进行描述而定义的
- 定义数据类型是为了把数据分成所需内存大小不同的数据，以便充分利用内存

- 不同的类型分配不同的长度和存储形式。因此，数据类型就是对数据分配存储单元的安
排，包括数据的存储形式以及编译系统为其
分配的存储单元的长度（字节）
- 综上，数据类型是一个值的集合以及定义在
这个值集上的一组操作

C语言中的数据类型



数据类型与程序的可移植性

- 数据类型与计算机系统结构密不可分
 - ✓ 16位机器：int、short型变量都用2个字节表示
 - ✓ 32位机器：int型变量用4个字节表示，short则为2个字节
- 为增加程序的可移植性，定义变量时需斟酌
 - ✓ 尽量使用标准库函数
 - ✓ 尽可能使程序适用于所有的编译器
 - ✓ 使用#ifdef指令将不可移植的代码分离出来

数据类型与计算精度

- 选择合适的数据类型，避免数据溢出

```
#include<stdio.h>
void main( ) {
    short int a=32767, b;
    b=a+1;
    printf("a=%d b=%d\n", a, b);
}
```

a=32767 b=-32768

- 为减少舍入误差，定义变量时要考虑有效数字
 - ✓ float: 7位有效数字
 - ✓ double: 15位有效数字
 - ✓ long double: 19位有效数字

程序设计中数据的组织方式

- 一般数据的组织
- 批量数据的组织
- 混合数据的组织
- 动态数据的组织
- 磁盘数据的组织

3.2 一般数据的组织与处理

1. 整型数据的表示

三种表现形式：

- **十进制整数**：正、负号，0~9，首位不是0
例：10, 123
- **八进制整数**：正、负号，0~7，首位是0
例：010, 0123
- **十六进制整数**：正、负号，0~9, a~f, A~F,
前缀是0x, 或 0X
例：0x10, 0X123 $123 \rightarrow 1111011_{(2)} \rightarrow 0173_{(8)} \rightarrow 0x7B_{(16)}$

最基本的整型类型

- 基本整型(int型): 占2个或4个字节
- 短整型(short int): VC++6.0中占2个字节
- 长整型(long int): VC++6.0中占4个字节
- 双长整型(long long int): C99新增, 8个字节

整型变量的符号属性

- 有符号基本整型 `[signed] int;`
- 无符号基本整型 `unsigned int;`
- 有符号短整型 `[signed] short [int];`
- 无符号短整型 `unsigned short [int];`
- 有符号长整型 `[signed] long [int];`
- 无符号长整型 `unsigned long [int]`
- 有符号双长整型 `[signed] long long [int];`
- 无符号双长整型 `unsigned long long [int]`

针对64位计
算机系统

2. 字符型数据

- 字符是按其代码（整数）形式存储的
- C99把字符型数据作为整数类型的一种
- 字符型数据在使用上有自己的特点

字符与字符代码

大多数系统采用ASCII字符集（附录B）

- 字母：A ~Z, a ~z
- 数字：0~9
- 专门符号：29个：! ” # & ‘ () *等
- 空格符：空格、水平制表符、换行等
- 不能显示的字符：‘\0’ (空(null)字符)、‘\a’ (警告)、‘\b’ (退格)、‘\r’ (回车)等

字符‘3’和整数3的：

- 字符‘3’只是代表一个形状为‘3’的符号，在需要时按原样输出，在内存中以**ASCII码（51）**形式存储，占1个字节

0	0	1	1	0	0	1	1
---	---	---	---	---	---	---	---

- 整数3是以整数存储方式（二进制补码方式）存储的，占2个或4个字节

0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

字符变量

- 用类型符char定义字符变量

```
char c = '?';
```

系统把“?”的ASCII代码63赋给变量c

```
printf("%d %c\n", c, c);
```

输出结果是: 

3. 浮点型数据

- 单精度浮点型 float
- 双精度浮点型 double
 - ◆ 编译系统为double型变量分配8个字节
 - ◆ 15位有效数字
- 长双精度型 long double

数据精度和取值范围

- 数据精度与取值范围是两个不同的概念：

```
float x = 1234567.89; 1234567.875000
```

//虽在取值范围内，但无法精确表达

```
float y = 1.2e55; 1. #INF00 //表示无穷大 (infinity 的缩写)
```

//y 的精度要求不高，但超出取值范围

- 不是所有浮点数都能在机器中精确表示

浮点型常量

- 浮点数的表示
 - 浮点表示法 : 0.123 123.4 12. .12
 - 科学计数法 : 1.23E-4 1.2e+3 1E-2
- 浮点数的类型

计算机中的浮点数都是double型, C编译系统把浮点型常量按双精度处理

4. 一般数据的输入输出

所谓输入输出是以计算机主机为主体而言的

◆ 从计算机向输出设备(如显示器、打印机等)

输出数据称为**输出**

◆ 从输入设备(如键盘、磁盘、光盘、扫描

仪等)向计算机输入数据称为**输入**

输入和输出操作是由库函数来实现的

- ◆ `printf`和`scanf` 是C语言的库函数的名字，用这两个函数时，必须指定其格式
- ◆ `getchar`、`putchar`、`gets`、`puts`
- ◆ 在使用输入输出函数时，要在程序文件的开头用预编译指令
`#include<stdio.h>` 或 `#include"stdio.h"` 或
`#include<string.h>`

printf函数

printf函数的一般格式

printf（格式控制，输出表列）

例如： `printf("i=%d, c=%c\n", i, c);`



输出格式

printf（格式控制，输出表列）

例如：

```
printf("i=%d, c=%c\n", i, c);
```

普通字符

可以是常量、变量或表达式

常用格式字符

- **d格式符**。输出有符号的十进制整数

- ◆ 可指定输出数据的域宽

```
printf("|%5d%5d|\n",12, -345);
```

```
| 12 -345 |
```

- ◆ %d输出int型数据

- ◆ %ld输出long型数据

- **c格式符**。用来输出一个字符

```
char ch='a';
```

```
printf("%c--%5c", ch, ch);
```

```
a--      a
```

- **s格式符**。用来输出一个字符串

```
printf("%s", "CHINA");
```

```
CHINA
```


• **f格式符**。用来以小数形式输出

```
double a=10;
```

```
printf("%f\n", a/7); 1.428571
```

- ◆ 用**%m.nf**指定数据宽度和小数位数

```
printf("%15.13f\n", a/7); 1.4285714285714
```

```
printf("%.0f\n", 1000/7.0); 143
```

- ◆ 用**%-m.nf**使输出的数据向左对齐

```
printf("|%-10.3f|\n", 1000/8.0); |125.000 |
```

- **e格式符**。指定以指数形式输出实数

- ◆ **%e**

- // VC++中小数位数为6位，指数部分占5列

- //小数点前必须有而且只有1位非零数字

- `printf("%e",12.345);` `1.234500e+001`

- ◆ **%m.ne**

- `printf("%11.2e",12.345);` `| 1.23e+001 |`

scanf函数

scanf 函数的一般形式

scanf(格式控制, 地址表列)

含义同printf函数

可以是变量的地址,
或字符串的首地址

scanf("%c%d%f",&x,&y,&z);

使用scanf函数时应注意的问题

`scanf("%d%d%d", a, b, c);`

`scanf("%d%d%d", &a, &b, &c);`

`scanf("x=%c,y=%d,z=%f", &x, &y, &z);`

a 1 2 ✓

```
a 1 2
x=-858993460z=-107374176.000000
```

x=a y=1 z=2 ✓

```
x=a y=1 z=2
x=ay=-858993460z=-107374176.000000
```

x=a,y=1,z=2 ✓

```
x=a, y=1, z=2
x=ay=1z=2.000000
```

`scanf("%c%c%c",&c1,&c2,&c3);`

A B C ✓

```
A B C
c1=A c2= c3=B
```

ABC ✓

```
ABC
c1=A c2=B c3=C
```

字符数据的输入输出

- 用 `getchar` 函数输入一个字符
 - 向计算机输入一个字符
 - 一般形式: `c=getchar();`
- 用 `putchar` 函数输出一个字符
 - 从计算机向显示器输出一个字符
 - 一般形式: `putchar(c);` 或 `putchar('A');`

字符串数据的输入输出

- 头文件 `string.h`
- 用 `gets` 函数输入一串字符
 - 向计算机输入一串字符
 - 一般形式: `gets(s);` // `s` 为字符数据名
- 用 `puts` 函数输出一串字符
 - 从计算机向显示器输出一串字符
 - 一般形式: `puts(s);` 或 `puts("Shanghai");`

3.3 批量数据的组织与处理

- 存放一个数据：1个变量
- 存放100个相同类型的数据：100个变量？

结论：基本数据类型已无法满足需要，

需利用构造数据类型之一：数组

认识数组

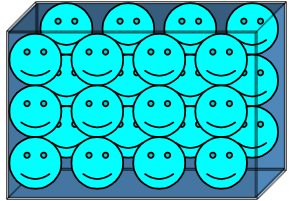
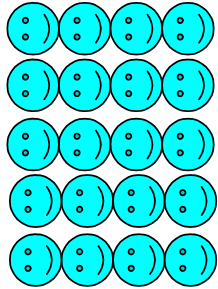
- 数组是变量的一个有序集合，其中所有变量具有同一类型。数组可以表示向量、矩阵、一行文字...
- 数组类型的基础类型：称为元素类型，可以是任意类型（基本类型、结构体/共用体类型、指针类型、数组类型...）

- 数组类型的构造方法：把固定数目的元素类型的数据顺序排成一个表格，每个数据是元素类型的一个值。所有元素值顺序排成的表示数组类型的一个值
- 数组类型的每个元素都有一个唯一的编号：称为“下标”

- 用数组名和下标区分不同的数组元素
- 如果数组元素又是一个数组，则该数组称为**多维数组**
- 多维数组用数组名和**多个下标**区分不同的元素

数组特点

- 相同**：构成数组的元素属于同一种类型
- 顺序**：数组元素按下标值从小到大顺序排列
- 连续**：下标相连的两个数组元素在内存中的位置相邻
- 静态**：系统为数组分配固定大小的空间，程序一旦运行，将无法更改数组所占空间大小和元素个数

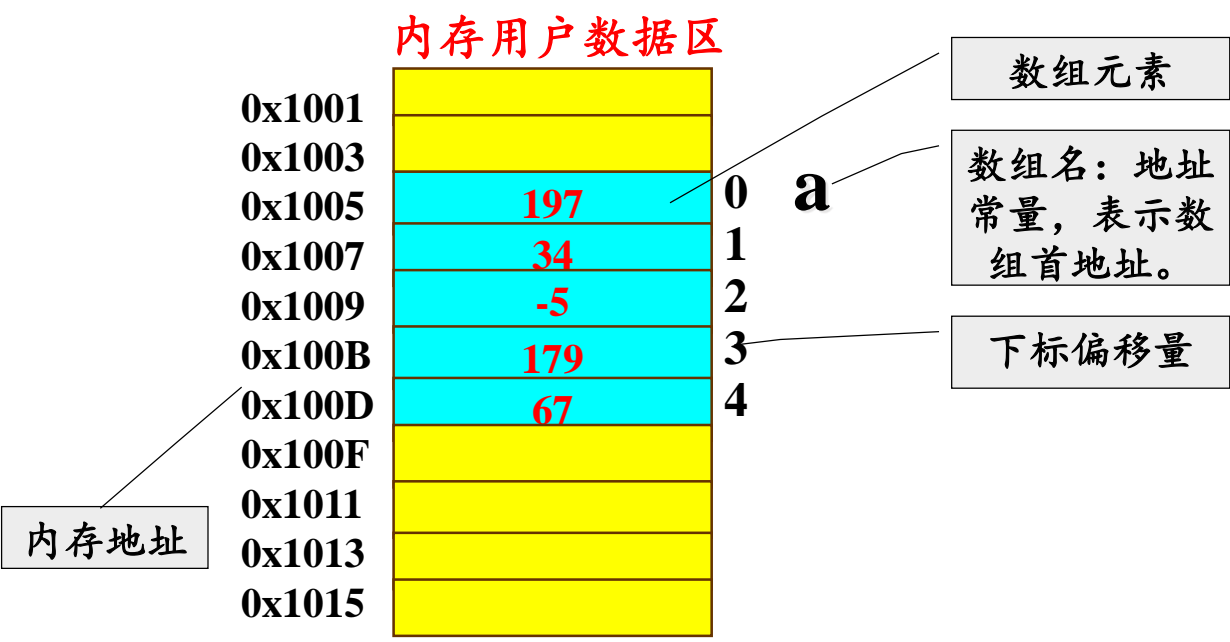


数组的操作

- 访问数组元素
- 遍历数组
- 删除一个数组元素
- 增加一个数组元素

C语言中的数组

例: `int a[5];`



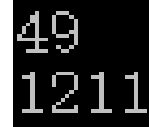
例： 将一任意十进制数转换成三进制并输出

分析：

用**整除3求余法**：1) 十进制数整除3得到的余数就是对应的三进制数的最低位，得到的商所对应的三进制数，恰好等于原三进制数去掉最低位后的剩余部分；2) 再用上述方法可得到三进制数的次低位；3) 反复上述过程即可得到三进制数的所有位。

三进制的各位数字用数组存放。

```
#include<stdio.h>
int main( ) {
    int i=0, x, d[20];
    scanf("%d", &x);
    while(x>0) {
        d[i++]=x%3;
        x=x/3;
    }
    for(i--; i>=0; i--)    printf("%d", d[i]);
    printf("\n");
    return 0;
}
```



49
1211

3.4 混合数据的组织与处理

- 对于大型数据集，通常采用数组来组织和处理。但数组要求在数据类型相同情况下才能使用
- 更多情况，需要处理的数据是一个对象或者是具有多种数据类型的一组信息。即一个数据项由多个子数据组成，每个子数据的类型可能不一样

- 对于这种情况，如果采用数组形式来组织这组信息，需要定义多个数组
- 且根据数组的存储形式，每一种属性是连续存储的，从而造成分配内存不集中、寻址效率不高的情况
- 显然，数组类型不适合用来组织混合类型数据
能否抽象或创建一种新的数据类型来组织这样的信息？

- 通常，采用表格来管理这类数据

字段 数据项

学号	姓名	性别	年龄	高考成绩	家庭住址
10001	张三	男	18	586	北京市朝阳区
10002	李四	女	19	595.5	上海市奉贤区
...					
30045	王五	男	17	589	四川省成都市

记录

C语言中混合数据的组织与处理

- C语言中，用户可自己建立由不同类型数据组成的组合型的数据结构，它称为**结构体**
- 例如，一个学生的学号、姓名、性别、年龄、高考成绩、家庭地址等项，是属于同一个学生的，因此组成一个组合数据，反映它们之间的内在联系

struct Stu

```
{ int num;  
  char name[20];  
  char sex;  
  int age;  
  float score;  
  char addr[30];  
};
```

- ◆ 由程序设计者创建了一个结构体类型 struct Stu
- ◆ 它包括 num、name、sex、age、score 和 addr 等不同类型的成员

- 前面只是建立了一个结构体类型，并没有定义变量，且无具体数据，系统对之不分配存储单元
- 为了能在程序中使用结构体类型的数据，应当定义结构体类型的变量，并在其中存放具体的数据

用所创建的数据类型定义**结构体变量**

struct Stu student1, student2;

结构体类型名

结构体变量名

student1

10001	张三	M	18	586	北京市朝阳区
-------	----	---	----	-----	--------

student2

10002	李四	F	19	595.5	上海市奉贤区
-------	----	---	----	-------	--------

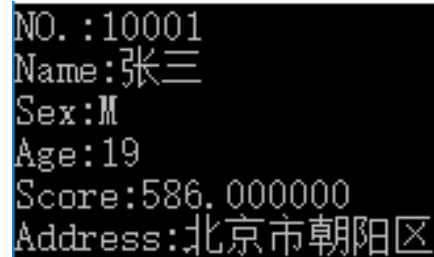
例：编写程序输出一个学生的相关信息，
包括：学号、姓名、性别、年龄、高考成绩、家庭住址

• **解题思路：**

- 自己创建一个结构体类型，包括有关学生信息的各成员
- 用它定义结构体变量，同时赋以初值
- 输出该结构体变量的各成员


```
#include <stdio.h>
struct Stu
{
    long int num;
    char name[20];
    char sex;
    int age;
    float score;
    char addr[20];
}student={10001,"张三",'M', 19, 586, "北京市朝阳区"};
```

```
int main( )  
{  
    printf("NO.:%ld\nName:%s\nSex:%c\nAge:%d\n  
        Score:%f\nAddress:%s\n", student.num,  
        student.name, student.sex, student.age,  
        student.score, student.addr);  
    return 0;  
}
```

A screenshot of a terminal window showing the output of the C program. The text is displayed on a black background with white characters. The output consists of six lines: 'NO. :10001', 'Name:张三', 'Sex:M', 'Age:19', 'Score:586.000000', and 'Address:北京市朝阳区'.

```
NO. :10001  
Name:张三  
Sex:M  
Age:19  
Score:586.000000  
Address:北京市朝阳区
```

3.5 动态数据的组织与处理

计算机程序内存的分配与管理方式

- **编译时分配**：在编译阶段由操作系统负责分配及撤销，程序员不能干预。**静态分配**
- **运行时分配**：程序员根据需要为变量等分配需要大小的存储空间，并可根据程序的运行情况撤销这些空间。**动态分配**

- 普通变量和数组属于静态数据结构
 - ◆ 变量在程序执行时所占内存空间大小是固定的
 - ◆ 定义数组时，数组长度必须是常量，系统会为数组元素分配一段连续的内存空间。因此，如何确定数组长度非常关键，既要能放得下数据，又不要浪费存储空间
- 例：存储计算机2018级“计算机程序设计”教学班高于平均成绩的学生的相关信息 ...

- **动态数据结构**——程序执行过程中根据需要来扩展和收缩的数据结构
 - ◆ 动态数据结构在程序中没有“显式”定义，它没有名字，在编译时程序不知道有该数据结构，不给其分配内存空间
 - ◆ 提高了空间利用率：内存随时按需分配和释放
 - ◆ 申请实在程序运行过程中进行的
 - ◆ 对已申请的空间，可以改变其大小

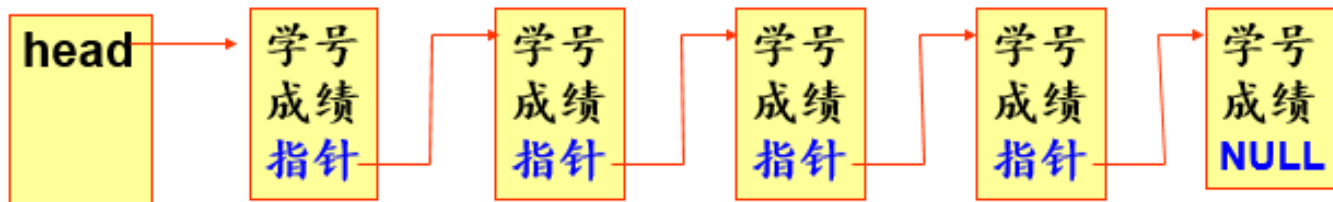
C语言中的动态数据组织与处理

(1) 动态内存处理

- ✓ malloc() //分配一定字节的连续内存空间
- ✓ calloc() //分配若干个具有固定字节的连续内存空间
- ✓ realloc() //修改已分配的内存区的大小
- ✓ free() //释放已分配的内存区

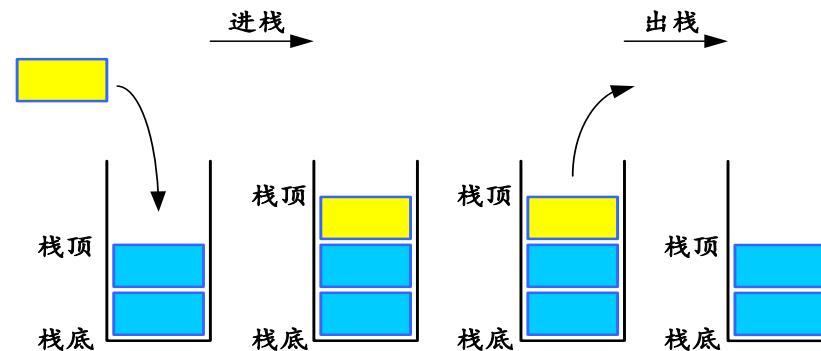
(2) 链表

- ◆ 一种物理存储单元上非连续、非顺序的存储结构，数据元素通过链表中的指针链接次序实现的。链表由一系列结点组成，结点可以在运行时动态生成
- ◆ 每个结点包括两个部分：一个是存储数据元素的数据域，另一个是存储下一个结点地址的指针域



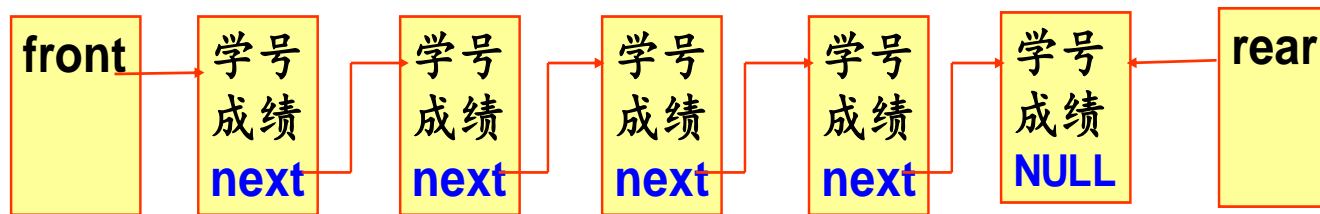
(3) 栈

- ◆ stack，一种运算受限的线性表，仅允许在表的一端（栈顶）进行插入和删除运算，另一端为栈底
- ◆ 插入新元素到栈顶元素的上面，称作进栈、入栈或压栈；从一个栈删除元素又称作出栈或退栈



(4) 队列

- ◆ 一种操作受限制的特殊线性表
- ◆ 只允许在表的前端（front，队头）进行删除操作，而在表的后端（rear，队尾）进行插入操作
- ◆ 队列中没有元素时，称为空队列



3.6 磁盘数据的组织与处理

- 实际问题的解决方案中往往包含大量数据，这些数据可能是由程序生成的输出数据，也可能是程序所需的输入数据
- 无论输入数据或输出数据，当数据规模较小时，尚可满足需求
- 但数据量较大时，如果直接输入，时间上不允许，且一旦输错，则需要全部重新输入；如果直接输出则可能无法在显示器上全部显示出来

- 当一个计算机程序运行时，所有数据信息（常量、变量、数组、结构体变量）都临时存储在内存中，一旦程序运行结束，分配给程序的内存将被操作系统收回，这些信息将会消失。如果下次想利用这些数据...？
- 但如果将这些数据信息以文件的形式存储到外部介质（磁盘、U盘、光盘等）上，长期保存起来，上述问题将迎刃而解

- 以文件方式进行数据的组织与处理，文件内容可以被传输，也可以在随后被自己或其他程序读取（输入）。输出时，对于大批量数据以文件形式存放到外部介质上，方便随时查看和进行数据处理

文件是计算机程序设计过程中组织和处理大规模数据的有效方法

数据类文件的分类

- (1) 按照数据结构来组织、存储和管理数据的仓库，称为“**数据库**”。由数据库管理系统按记录或字段存取数据集
- (2) 不带有数据结构的数据文件，文件中的数据是一串字节，没有结构，称为“**流式文件**”，简称“**文件**”。计算机程序设计语言中提供了对流式文件操作的方法，运行效率较高

C语言对外部数据的组织和处理

- 对流式文件，可按ASCII码和二进制码存储，称为“**文本文件**”和“**二进制文件**”
- 文件使用前需先打开，用完后需关闭
- 对数据文件，以**随机方式**或**顺序方式**进行读写
- 程序中使用的每一个数据文件都有一个与之关联的**文件指针**，通过文件指针对文件内容进行具体的操作

例：编写程序输入一串文字信息保存到磁盘上，直到输入字符为“@”为止。然后打开锁创建的文件，读出文件内容并显示到屏幕上

• **解题思路：**

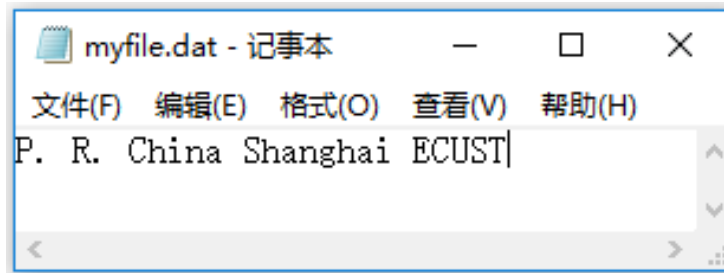
- 自己创建一个文件
- 从键盘逐个输入字符，然后用fputc函数写到磁盘文件
- 用fgetc函数从文件中读出信息显示到显示器上

```
#include <stdio.h>
#include<stdlib.h>
void main( )
{
    FILE *fp;
    char ch;
    if((fp=fopen('f:/CAI/C/tmp/myfile.dat','w'))==NULL)
        { printf('打开文件失败\n');  exit(0);  }
    while((ch=getchar())!='@')  fputc(ch, fp);
    fclose(fp);
```



```
if((fp=fopen("f:/CAI/C/tmp/myfile.dat","r"))==NULL)
    { printf("打开文件失败\n");  exit(0); }
while((ch=fgetc(fp))!=EOF) putchar(ch);
printf("\n");
fclose(fp);
}
```

```
P. R. China Shanghai ECUST@
P. R. China Shanghai ECUST
```



Homework

实践

实验3（见课程网站系统）

要求：提交纸质实验报告

作业

《教材》： P82-83 3、6、7