计算机程序设计

Computer Programming

期中小结

主讲：吴锋

目录
CONTENTS

良好的编码风格

- 良好的编码风格能够提高代码的可读性，提高编码效率，减少Bug的数量。
- 良好的编码风格能够提高代码的可维护性，有利于团队合作，以及持续的改进。
- 不同的编码风格没有优劣之分，重要的是保持代码风格的一致性，符合团队规范。

◎ 标识符的命名

- 匈牙利命名法为C程序标识符的命名定义了一种非常标准化的方式。这种命名方式是以两条规则为基础的：
  - 变量的名字以一个或者多个小写字母前缀开头，前缀能够体现变量数据类型、作用域等信息。
  - 在标识符内，前缀以后就是一个或者多个第一个字母大写的单词，这些单词清楚地指出了该标识符的作用。

◎ 标识符的命名

- 变量命名加前缀表面变量的类型

n	int
c	char
f	浮点数 (float)
b	取值只为真和假的整型变量 如 bValid
p	指针
a	数组
fp	文件指针 FILE *

◎ 标识符的命名

- 变量名中单词开头字母大写或以‘_’隔开，其他字母小写
  - 如：bValid 或 b_valid
  - 但是常用的意义明显的变量，如 i,j,k, 坐标 x,y等不必遵循 1),2)
- 常量和宏都是大写，单词间用 ‘_’分隔

```
#define      MAX_WIDTH   5
#define      ABS(x)      ((x)>=0?(x):-(x))
```

◎ 标识符命名基本原则

- 标识符号应能提供足够信息，最好是可以发音的。
- 为全局变量取长的描述信息多的名字，为局部变量取短名字。
- 名字太长时可以适当采用单词的缩写。但要注意，缩写方式要一致。要缩写就全都缩写。
  - 如：单词Number, 如果在某个变量里缩写成了: int nDoorNum；那么最好包含 Number单词的变量都缩写成 Num。
- 注意使用单词的复数形式。
  - 如：int nTotalStudents, nStudents ;容易让人理解成代表学生数目，而 nStudent 含义就不十分明显。
- 一般变量和结构名用名词，函数名用动词或动宾词组。

◎ 标识符命名基本原则

- 对于返回值为真或假的函数，加“Is”前缀如：

```
int         IsCanceled();
int         isalpha(); // C语言标准库函数
BOOL        IsButtonPushed();
```

- 对于获取某个数值的函数，加“Get”前缀

```
char * GetFileName();
```

- 对于设置某个数值的函数，加“Set”前缀

```
void SetMaxVolume();
```

程序书写格式

- 正确使用缩进：
  - 首先，一定要有缩进，否则代码的层次不明显。
  - 缩进应为4个空格较好。需要缩进时一律按Tab键，或一律按空格键，不要有时用Tab键缩进，有时用空格键缩进。
  - 一般开发环境都能设置一个Tab键相当于多少个空格，此时就都用Tab键。

程序书写格式

- 行宽与折行：
  - 一行不要太长，不能超过显示区域。以免阅读不便。
  - 太长则应折行。折行最好发生在运算符前面，不要发生在运算符后面如：

```
if (Condition1() && Condition2()
        && Condition3()) {
}
```

◎ 程序书写格式

- 注意 ‘{’,‘}’位置不可随意，要统一。
  - 如果写了：

```
if (condition1()) {
        DoSomething();
}
```

  - 别处就不要写：

```
if (condition2())
{
        DoSomething() ;
}
```

◎ 程序书写格式

- 变量和运算符之间最好加1个空格

```
int  nAge = 5;
nAge = 4;
if (nAge >= 4)
        printf( “%d”,nAge);

for (i = 0; i < 100; i++);
```

一些编程习惯

- 尽量不要用立即数，而用const 定义成常量或定义成宏，以便以后修改。

```
const  int MAX_STUDENTS  = 20
struct SStudent aStudents [MAX_STUDENTS];
```

比

```
struct SStudent aStudents [20];
```

好

```
#define TOTAL_ELEMENTS  100
for(  i = 0; i < TOTAL_ELEMENTS; i ++)   { }
```

一些编程习惯

- 使用sizeof()运算符，不直接使用变量所占字节数的数值，以便适应不同的系统

```
int nAge;
for( j = 0; j < 100; j++ )
        fwrite( fpFile,& nAge, 1, sizeof(int));
```

比

```
for( j = 0; j < 100; j++ )
        fwrite( fpFile,& nAge, 1, 4);
```

好

一些编程习惯

- 稍复杂的表达式中要积极使用括号，以免优先级理解上的混乱

```
n = k +++ j;  //不好
n = ( k ++ ) + j; //好一点
```

- 不很容易理解的表达式应分几行写：

```
n = ( k ++ ) + j; 应该写成：
n = k + j;
k ++;
```

一些编程习惯

- 不提倡在表达式中使用？：形式，而用if .. else语句替代。

```
xp =  2 * k < ( n-m) ? c[k+1] : d[k--];
```

```
if( 2*k < (n-m))
        xp = c[k+1];
else
        xp = d[k--];
```

一些编程习惯

- 嵌套的if else 语句要多使用 { }

```
if( Condition1() )
        if( Condition2() )
                DoSomething();
        else
                NoCondition2();
```

不够好，应该：

```
if( Condition1() ) {
    if( condition2() )
        DoSomething();
    else
        NoCondition2();
}
```

一些编程习惯

- 应避免 if else 多重嵌套，而用并列完成。

```
if( Condition1() ) {
    if ( Condition2() ) {
            if( Condition3() ) {

            Condition123();
            }else  {

            NoCondition3();
            }
    }else {
            NoCondition2();
    }
}else {
    NoCondiction1();
}
```

替换为：

```
if( ! condition1 ) {
    NoCondition1();
}else if( ! condition2 ) {
    NoCondition2();
}else if( ! condition3) {
    NoCondition3();
}else {
    Condition123();
}
```

一些编程习惯

- 写出来的代码应该容易读出声

比如

```
if( !( n > m ) && !( s > t))
```

就不如

```
if( ( m <= n ) && ( t <= s ))
```

```
if( !( c == 'y' || c == 'z'))
```

不如

```
if( c!= 'y' && c!= 'z');
```

输入输出函数

- 非格式化输入输出：
  - getchar / putchar： 输入 / 输出一个字符
  - gets (gets_s) / puts： 输入 / 输出一个字符串
- 格式化输入输出：
  - scanf / printf：从键盘输入 / 输出到终端
  - sscanf / sprintf：从字符串输入 / 输出到字符串
  - fscanf / fprintf：从文件输入 / 输出到文件
- 所有输入输出都必须包含头文件 stdio.h

```
#include <stdio.h>
```

◎ 非格式化输入输出

- 输入输出一个字符
  - int getchar(void); // 成功时返回一个字符，失败时返回 EOF
  - int putchar(int ch); // 成功时返回输出的字符，失败时返回 EOF
- 输入输出一个字符串
  - char *gets(char *str); / char *gets_s(char *str, rsize_t n);
    - 当遇到换行'\n'或文件末尾时，从终端读入字符串到str （不包含'\n'）
    - 成功时返回str指针，失败时返回NULL指针。
  - int puts(const char *str);
    - 输出字符串str （并附带换行 '\n'），字符末尾的 '\0' 不会输出
    - 成功时返回一个非负值（与编译器相关），失败时返回 EOF

◎ 非格式化输入输出

- 输入输出一个字符
  - int getchar(void); 等价于 int fgetc(stdin);
  - int putchar(int ch); 等价于 int fputc(int ch, stdout);
- 输入输出一个字符串
  - char *gets(char *str); 对字符串数字越界没有检测，容易受到溢出攻击，因此不安全。通常建议使用 char *gets_s(char *str, rsize_t n); 或者 char *fgets(char *str, int count, stdin);
  - int puts(const char *str); 会在输出字符串的末尾自动添加换行'\n'，而int fputs(const char *str, stdout); 不会输出换行。
  - puts / fputs 的返回值有些编译器是输出字符长，有些是最后一个字符，甚至有些是任意非负值。

◎ 格式化输入输出

int **scanf**(const char *format , ...);

- 参数可变的函数
  - 第一个参数是格式字符串，后面的参数是变量的地址，函数作用是按照第一个参数指定的格式，将数据读入后面的变量
- 返回值
  - >0： 成功读入的数据项个数；
  - 0 ： 没有项被赋值；
  - EOF：第一个尝试输入的字符是EOF(结束)

◎ 格式化输入输出

int **printf**(const char *format , ...);

- 参数可变的函数
  - 第一个参数是格式字符串，后面的参数是待输出的变量，函数作用是按照第一个参数指定的格式，将后面的变量在屏幕上输出
- 返回值
  - 成功打印的字符数；
  - 返回负值为出错

◎ 格式控制符号

%d 读入或输出int变量

%c 读入或输出char变量

%f 读入或输出float变量

%s 读入或输出char * 变量

%lf 读入或输出double 变量

%e 以科学计数法格式输出数值

%x 以十六进制读入或输出 int 变量

%p 输出指针地址值

%.5lf 输出浮点数，精确到小数点后5位

◎ 格式化输入输出（例）

```
int n = scanf("%d%c%s%lf%f", &a, &b, c, &d, &e); // 输入是变量的地址，除数组、指针外一般变量要用取地址符
printf("%d %c %s %lf %e %f %d",a, b, c, d, e, e, n);
```

Input:

123a teststring 8.9 9.2

Output:

123 a teststring 8.900000 9.200000e+000 9.200000 5

Input:

123ateststring 8.9 9.2

Output:

123 a teststring 8.900000 9.200000e+000 9.200000 5

Input:

123 a teststring 8.9 9.2

Output:

123 a 0.000000 0.000000e+000 0.000000 3

正确的输入语句：

```
int n = scanf("%d %c%s%lf%f", &a, &b, c, &d, &e);
```

◎ 格式化输入输出

- **scanf** 常见错误
  - 输入项必须是变量的地址。
    - 如：char str[10]; scanf("%c %s", &str[0], str);
  - 若输入格式串中加入了格式符以外的其它字符，则输入时必须同样输入，否则结果不确定；
  - 若无其它字符，则可用空格、回车或制表符作为间隔标记。
  - 使用“%c”时，输入的字符不能加间隔标记。
  - 输入数据遇到空格、回车或制表符以及其它各种非法输入时，认为该项数据输入结束。注意，这些输入并没有被跳过去。
  - 输入数据比表列更多时，会被自动用于下一次输入。

◎ 格式化输入输出

- **printf** 常见错误
  - 输出项表列需要与格式控制字符一一对应
  - 类型不匹配时，不会进行隐式类型转换，会解析失败，输出0
  - 当格式符个数少于输出项时，多余的输出项不予输出。
  - 当格式符个数多于输出项时，结果为不定值。

◎ 格式化输入输出

int **sscanf**(const char *buffer, const char * format[, address, ...]);

- 和scanf的区别：从字符串buffer里读取数据

int **sprintf**(char *buffer, const char *format[, argument, ...]);

- 和printf的区别：往字符串buffer里输出数据

int **fscanf**(FILE *fp, const char * format[, address, ...]);

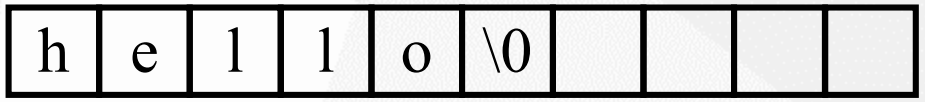
- 和scanf的区别：从文件fp里读取数据

int **fprintf**(FILE *fp, const char *format[, argument, ...]);

- 和printf的区别：往文件fp里输出数据

字符串

- 每个字符串是一个特殊的数组，满足两个条件
  - 元素的类型为char
  - 最后一个元素的值为‘\0’, Ascii码就是 0
- 以字符型数组存储
  - 从0号元素开始存储
  - 最大可以存储长度为N-1的字符串，N是数组的大小。
  - 例：字符串“hello”在长度为10的字符串数组中的存储



◎ 字符串处理函数

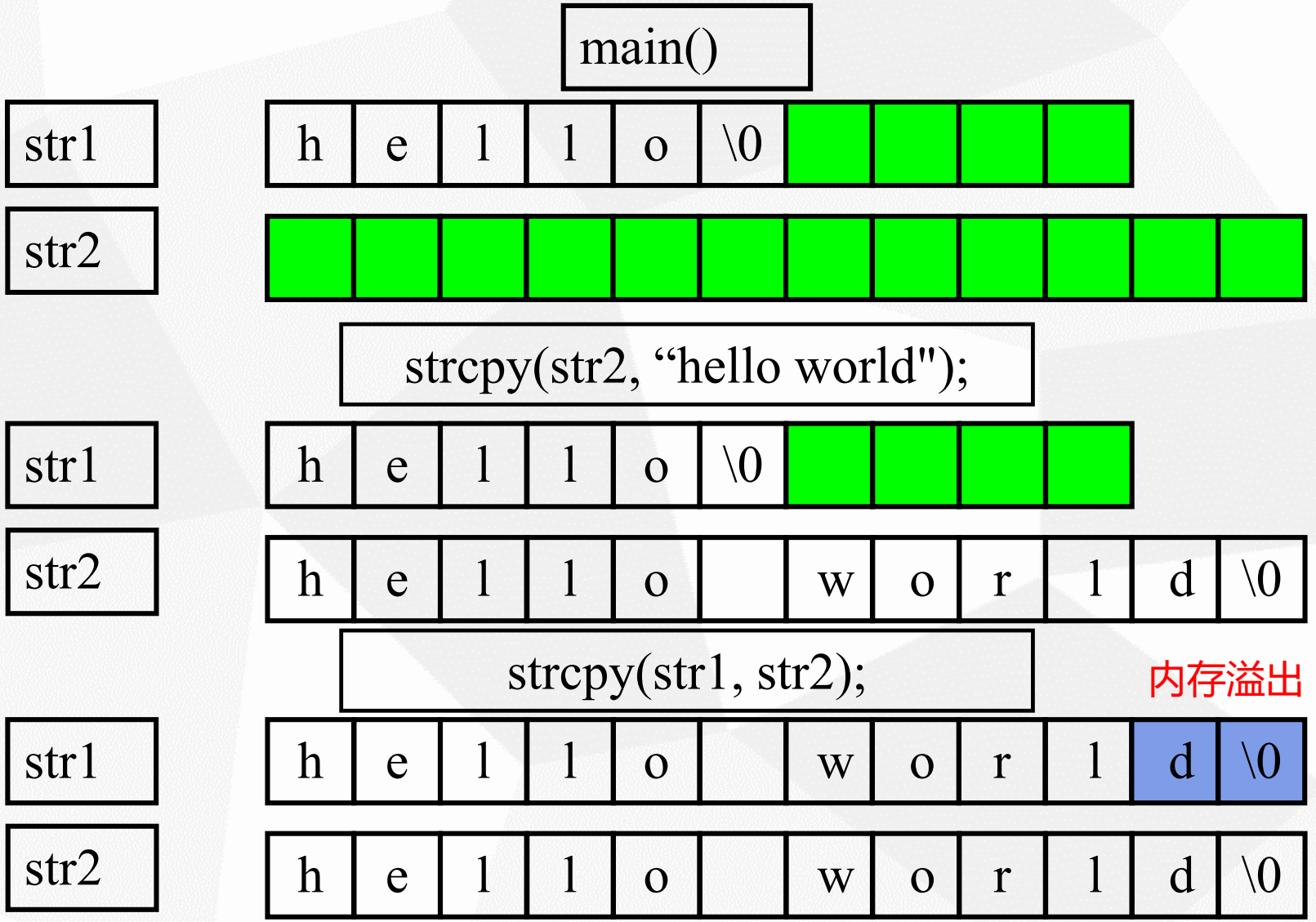
- 将格式化数据写入字符串： sprintf
- 字符串长度查询函数： strlen
- 字符串复制函数： strcpy、strncpy
- 字符串连接函数： strcat
- 字符串比较函数： strcmp、strncmp、stricmp、strnicmp
- 字符串搜索函数： strcspn、strspn、strstr、strtok、strchr
- 字符串大小写转换函数： strlwr、strupr
- 这些函数都要求 #include <string.h>

◎ 字符串处理函数

```
int my_strchr(char * s,  char c) { //看s中是否包含 c
        for( int i = 0; i < strlen(s) -1 ; i ++ )
                if( s[i] ==  c)
                        return 1;
        return 0;
}
```

- 有什么问题？这个函数执行时间和 s 的长度是什么关系？
- 问题：strlen 是一个**O(N)**的函数，每次判断 i < strlen(s) – 1 都要执行，太浪费时间了。

◎ 字符串拷贝


main()
str1
h e l l o \0
str2
strcpy(str2, “hello world");
str1
h e l l o \0
str2
h e l l o w o r l d \0
strcpy(str1, str2);
内存溢出
str1
h e l l o w o r l d \0
str2
h e l l o w o r l d \0

◎ 字符串的比较

- int **strcmp**(const char *s1, const char *s2);
  - 区分字符串中字符的大小写
  - 字符串根据字典序
    - 如果s1 < s2，则返回值 < 0
    - 如果s1 == s2，则返回值 == 0
    - 如果s1 > s2，则返回值 > 0
- int **stricmp**(const char *s1, const char *s2);
  - 不区分字符串中字符的大小写
  - 返回值情况与strcmp相同

◎字符串的查找

- 查找子串： char ***strstr**(char *s1, char *s2);
  - 查找给定字符串在字符串中第一次出现的位置。
  - 如果找到，则返回一个指向s1中第一次出现s2的位置的指针，及子串的起始地址。
  - 如果找不到，则返回 NULL。
- 查找字符： char ***strchr**(char *s, int c);
  - 查找给定字符在字符串中第一次出现的位置。
  - 如果找到，则返回一个指向s1中第一次出现c的位置的指针，及子串的起始地址。
  - 如果找不到，则返回 NULL。

◎ 字符串的部分拷贝

- 部分拷贝： char ***strncpy**(char *dest, char *src, int maxlen);
  - 将前 maxlen 个字符从src拷贝到dest。
  - 如果src中字符不足 maxlen 个，则连’\0’一起拷贝，’\0’后面的不拷贝。
  - 如果src中字符大于等于maxlen个，则拷贝 maxlen个字符。

◎ 字符串举例：子串

- 问题描述
  - 给出一些由英文字符组成的大小写敏感的字符串的集合S，请找到一个最长的字符串X，使得对于S中任意字符串Y，X或者是Y的子串，或者X中的字符反序之后得到的新字符串是Y的子串。要求写程序，输出X的长度。
- 例如：
  - 输入：S = {ABCD, BCDFF, BRCD}, X=CD，输出：2
  - 输入：S = {rose, orchid}, X=ro, 输出：2

◎ 字符串举例：子串

- 思路：
  - 随便拿出输入数据中的一个字符串。
  - 从长到短找出它的所有子串，直到找到否符合题目要求的。
- 改进：
  - 不要随便拿，要拿输入数据中最短的那个。
  - 从长到短找出它的所有子串，直到找到否符合题目要求的。

◎ 字符串举例：子串

```
#include <stdio.h>
#include <string.h>
int t, n;   char str[100][101];
int searchMaxSubString(char * source);
int main()  {
        int i, minStrLen, subStrLen; char minStr[101]; scanf("%d", &t);
        while (t--) {
                scanf("%d", &n); minStrLen = 100;  //记录输入数据中最短字符串的长度
                for (i = 0; i < n; i++) { //输入一组字符串
                        scanf("%s", str[i]);
                        if ( strlen(str[i]) < minStrLen ) { //找其中最短字符串
                                strcpy(minStr, str[i]);
                                minStrLen = strlen(minStr);
                        }
                }
                subStrLen = searchMaxSubString(minStr); //找答案
                printf("%d\n", subStrLen);
        }
        return 0;
}
```

◎ 字符串举例：子串

```
int searchMaxSubString(char * source) {
        int subStrLen = strlen(source),   sourceStrLen = strlen(source);
        int i, j;  int foundMaxSubStr; char subStr[101], revSubStr[101];
        while ( subStrLen > 0 ) { //搜索不同长度的子串，从最长的子串开始搜索
                for (i = 0; i <= sourceStrLen - subStrLen; i++) {
                        //搜索长度为subStrLen的全部子串
                        strncpy(subStr, source+i, subStrLen);
                        strncpy(revSubStr, source+i, subStrLen);
                        subStr[subStrLen] = revSubStr[subStrLen] = '\0';
                        _strrev(revSubStr);   foundMaxSubStr = 1;
                        for ( j = 0; j < n; j++)  //遍历所有输入的字符串
                            if  ( strstr(str[j], subStr) == NULL && strstr(str[j], revSubStr) == NULL ) {
                                    foundMaxSubStr = 0; break;
                            }
                        if (foundMaxSubStr)  return subStrLen;
                }
                subStrLen--;
        }
        return 0;
}
```

期中复习要求

- 阅读贾书《第三部分 上机实验》中的实验一至实验六
  - 掌握其中的“要点综述”部分的内容
  - 完成其中的“实验操作”部分的范例和实验内容

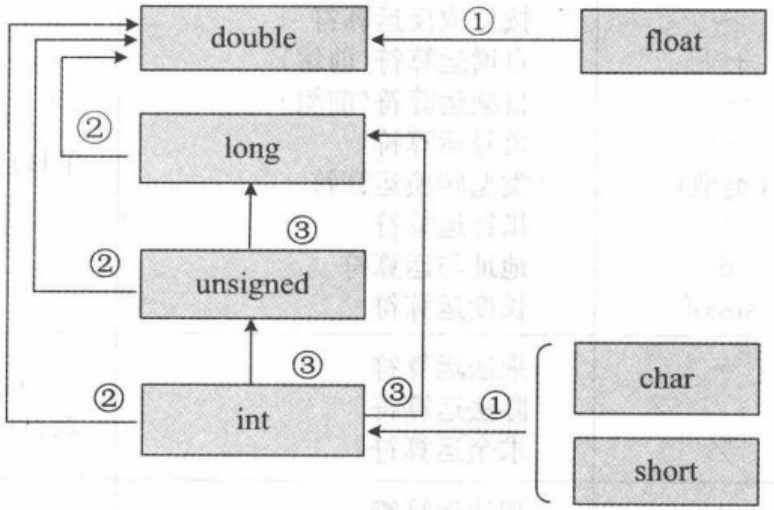


图 3.6 数据类型间的转换

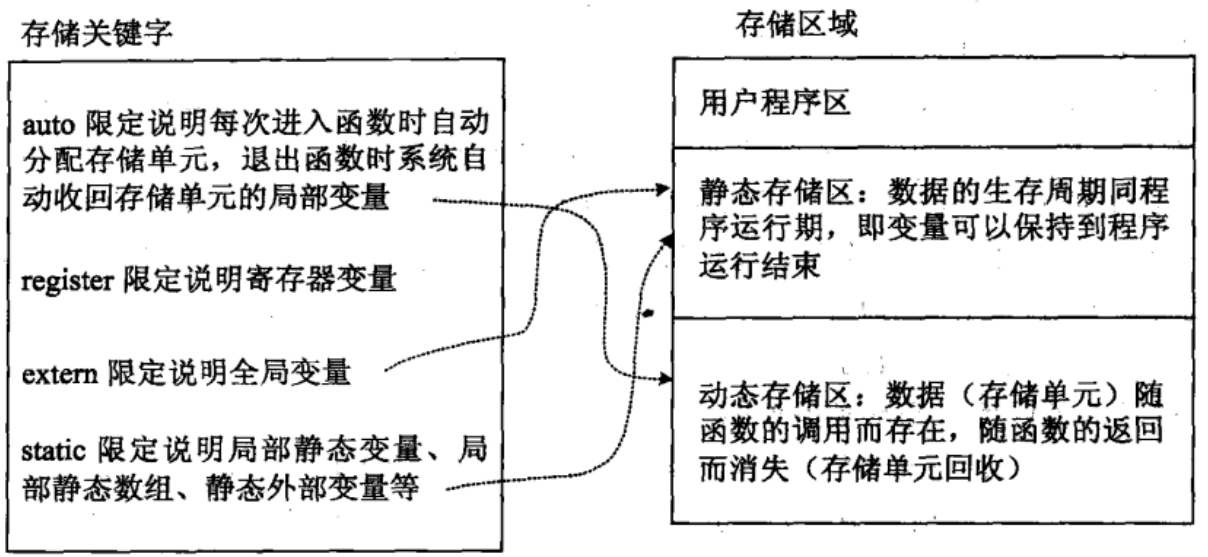


图 3.11 存储类别关键字

◎求方程的根：牛顿迭代法（P95）

2. 用牛顿迭代法求 $f(x)=x^3-2x^2+4x+1=0$ 在 x_0 附近的一个实根。

用牛顿迭代法求 $f(x)=0$ 在 x_0 附近的一个实根的方法是：

(1) 选一个接近于 x 的真实根的近似根 x_1。

(2) 通过 x_1 求出 $f(x_1)$。在几何上就是作 $x=x_1$，交 $f(x)$ 于 $f(x_1)$。见图 3.7。

(3) 过 $f(x_1)$ 作 $f(x)$ 的切线，交 x 轴于 x_2。可以用公式求出 x_2。由于

$$f'(x_1)=\frac{f(x_1)}{x_1-x_2}$$

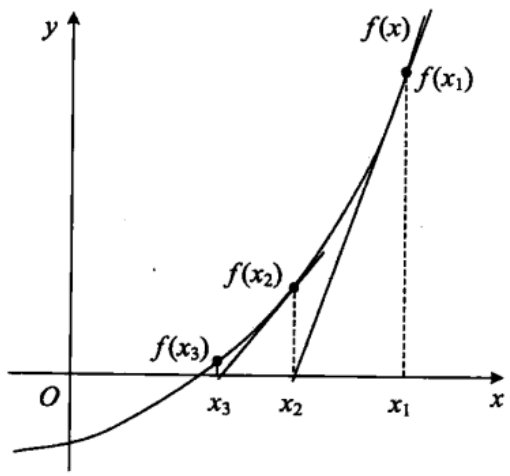


图 3.7　用牛顿迭代法示意图

故

$$x_2=x_1-\frac{f(x_1)}{f'(x_1)}$$

(4) 通过 x_2 求出 $f(x_2)$。

(5) 再过 $f(x_2)$ 作 $f(x)$ 的切线交 x 轴于 x_3。

(6) 通过 x_3 求出 $f(x_3)$，

……

一直求下去，直到接近真正的根，当两次求出的根之差 $|x_{n+1}-x_n|\leqslant\varepsilon$（$\varepsilon$ 一般可取 10^{-6}）就认为 x_{n+1} 足够接近于真实根。

牛顿迭代公式是：

$$x_{n+1}=x_n-\frac{f(x_n)}{f'(x_n)}$$

本例已知：$f(x)=x^3-2x^2+4x+1$

可求得：$f'(x)=3x^2-4x+4$

```
#include <stdio.h>
#include <math.h>
main(){
    double x,x1,f,f1;
    int n=1;
    scanf("%lf",&x);
    do{
        x1=x;
        f=pow(x1,3.0)-2.0*pow(x1,2.0)+4.0*x1+1;
        f1=3.0*pow(x1,2.0)-4.0*x1+4.0;
        x=x1-f/f1;
        printf("n=%4dx1=%-12.7fx=%-12.7f\n",n++,x1,x);
    }while(fabs(x-x1)>=1e-6);
    printf("root=%-12.7f\n",x);
}
```

运行记录：

```
1.0
n=   1x1=1.0000000    x=-0.3333333
n=   2x1=-0.3333333   x=-0.2287582
n=   3x1=-0.2287582   x=-0.2225152
n=   4x1=-0.2225152   x=-0.2224945
n=   5x1=-0.2224945   x=-0.2224945
root=-0.2224945
```

◎求方程的根：二分法（P98）

3. 用二分法求 $f(x)=x^3-6x-1=0$ 在 $x=2$ 附近的一个实根。

二分法的基本思路为：任取两点 x_1 和 x_2，判断 (x_1,x_2) 区间内有无一个实根。如图 3.8 所示，如果 $f(x_1)$、$f(x_2)$ 符号相反，说明 (x_1,x_2) 之间有一实根。取 (x_1,x_2) 的中点 x_0，检查 $f(x_0)$ 与 $f(x_1)$ 是否同符号，如果不同号，说明实根在 (x_0,x_1) 区间，这样就已经将寻根的范围减少了一半。然后用同样的办法再进一步缩小范围。再找 x_1 与 x_0 的中点，并且再舍弃其一半区间。如果 $f(x_0)$ 与 $f(x_1)$ 同号，则说明根在 (x_0,x_2) 区间，再取 x_0 与 x_2 的中点，并舍弃其一半区间。用这种方法不断缩小范围，直到区间相当小为止。

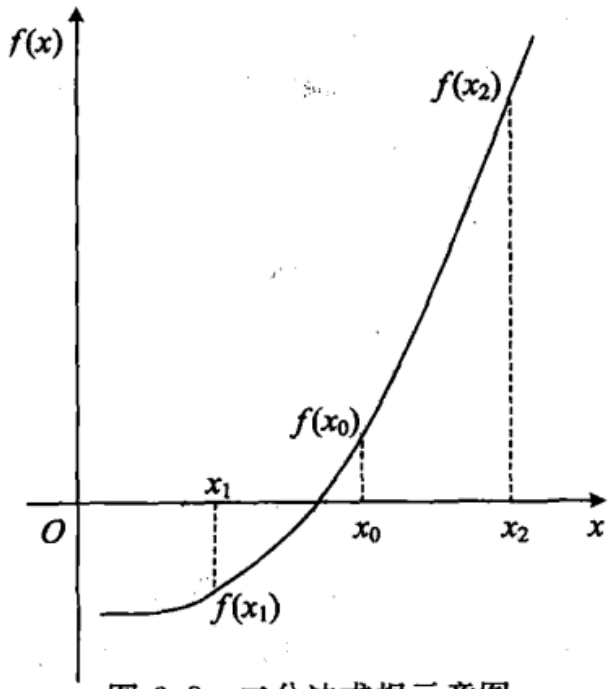


图 3.8　二分法求根示意图

一元二次方程 $ax^2+bx+c=0$ 的根。求方程解的公式如下：

$$x_{1,2}=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$$

```
#include "math.h"
main(){
  double x0,x1,x2,fx0,fx1,fx2;
  do{
  printf("enter x1,x2: ");
        scanf("%lf,%lf",&x1,&x2);
        fx1=pow(x1,3.0)-6*x1-1;
        fx2=pow(x2,3.0)-6*x2-1;
  }while(fx1*fx2>0);
  do{
       x0=(x1+x2)/2;
       fx0=pow(x0,3.0)-6*x0-1;
       if(fx0*fx1<0) {
            x2=x0;
            fx2=fx0;
       }
       else{
            x1=x0;
            fx1=fx0;
       }
  }while(fabs(fx0)>=1e-5);
  printf("root=%15.12lf\n",x0);
}
```

◎ 大作业

- 作业提交（任选1题完成）
  - 2022年1月1日前（发送至助教，建议提前完成，早提交早检查）
  - 文件名：PB21XXXXXX-姓名 大作业-数独.rar（报告、代码等）
  - Email名： PB21XXXXXX-姓名 大作业-数独
  - 按照贾书第五部分的格式撰写实验报告
- 评分标准
  - 每位学生独立完成
  - 项目质量：新颖度、代码量、算法难度、结论
  - 运行结果：可执行程序的运行结果
  - 代码风格：源码阅读，算法和风格检查
  - 编译结果：源码和可执行程序的一致性

实验报告格式

贾书《第五部分 实验报告》
P227-P234

中国科学技术大学

本科实验报告（模板）

课程名称 _____

实验名称 _____

姓　　名 _____

学　　院 _____

系　　别 _____

专　　业 _____

年　　级 _____

学　　号 _____

任课教师 _____

2018 年 11 月 27 日

一、如何写实验报告

针对“计算机程序设计”课程实验的特点，建议在书写实验报告时应包括如下内容：

1. 实验目的

实验作为教学的一个重要环节，其目的在于更深入地理解和掌握课程教学中的有关基本概念，应用基本技术解决实际问题，从而进一步提高分析问题和解决问题的能力，我们着手做一个实验的时候，必须明确实验的目的，以保证达到课程所指定的基本要求。在写实验报告时，要进一步确认是否达到了预期的目的。

2. 实验内容

在本书中，每一个实验都安排了多个实验题目，根据教学安排、进度、实验条件、可提供的机时、学生的基础等因素，可以选择其中的几个或全部。因此，在实验报告中，实验内容是指本次实验中实际完成的内容。在每一个实验题目中，一般都提出了一些具体要求，其中有些具体要求是为了达到实验目的而提出的。因此，在实验内容中，不仅要写清楚具体的实验题目，还要包括具体要求。

3. 算法与流程图

算法设计是程序设计过程中的一个重要步骤。在本书中，对于某些实验题目给出了方法说明，有的还提供了流程图。如果在做实验的过程中，使用的算法或流程图和书中的不一样，或者书中没有给出算法和流程图，则在实验报告中应给出较详细的算法说明与流程图，并对主要符号与变量作相应的说明。

4. 程序清单

程序设计的产品是程序，它应与算法或流程图相一致。程序要具有易读性，符合结构化原则。

5. 运行结果

程序运行结果一般是输出语句所输出的结果。对于不同的输入，其输出的结果是不同的。因此，在输出结果之前还应注明输入的数据，以便对输出结果进行分析和比较。

6. 调试分析和体会

这是实验报告中最重要的一项，也是最容易忽视的一项。

实验过程中大量的工作是程序调试，在调试过程中会遇到各种各样的问题，每解决一个问题就能积累一点经验，提高编程的能力。因此，对实验的总结，最主要的是程序调试经验的总结。调试分析也包括对结果的分析、尚存在的问题和拟解决的方法等。

体会主要是指通过本次实验是否达到了实验目的，有哪些基本概念得到了澄清等。

大作业1：作家书写习惯分析

- 实验目的
  - 编程实现若干著作的数据统计，分析多位作家的书写习惯
- 实验要求
  - 实现书籍内容的统计程序编写，设计统计指标，分析英文作家的书写习惯，如句子长短、单词长短、最喜欢用词等
  - 扩展分析：自选某个（或某些）角度分析小说。如分析男女作家写作风格的差异、写作语言、写作风格随着年代的变化等
  - 书写实验报告，图文并茂
- 实验内容
  - 编写读取、分析文章的程序（在网络上寻找作家及其作品资源）
  - 分析作品数据，得出结论

◎ 大作业2：矩阵相乘算法

- 实验目的
  - 编程实现Strassen矩阵相乘算法
- 实验要求
  - 基于C语言实现Strassen矩阵相乘算法；
  - 书写实验报告，图文并茂。
- 实验内容
  - 测试矩阵规模N=8、16、32、64、128、256、512乃至更高时的算法性能
  - 比较Strassen算法和传统$O(n^3)$算法的实际计算性能

大作业3：三种字符串匹配算法

- 实验目的
  - 学习/实现Brute Force、KMP、Aho-Corasick Automaton三种字符串匹配算法
- 实验要求
  - 利用C语言实现三种算法；
  - 书写实验报告，图文并茂。
- 实验内容
  - 编写代码，统计某本英文书籍中若干关键词出现频率。注意同一单词的大小写、时态，以及可能出现的单词子串仍是一个单词的情况（如'they'中有'the'）
  - 以图表的形式总结分析得到的结果

◎ 大作业4：数独

- 实验目的
  - 用C语言实现数独。
- 实验基本要求
  - 编写代码实现数独的部分功能，包括将当前数独状态输出到屏幕上、插入一个数、修改一个数、检查当前数独是否成立。
  - 数独库可以提前存储到文件中，利用文件操作进行读取。
- 加分项
  - 能够判断数独是否可解。
  - 能够给出当前状态下，数独的下一步的提示。
  - 能够进行题目的初始化，即能够让计算机给出一个最终只有唯一解的数独题目

大作业5：聊天机器人

- 实验目的
  - 用C语言实现简单的聊天机器人
- 实验基本要求
  - 编写代码实现聊天的基本功能，可与程序使用者用自然语言一问一答；
  - 聊天内容可以是通用，也可以是特定范围，比如，假定聊天机器人为科大学生，与人聊天科大的学习、生活情况
- 加分项
  - 应用机器学习
  - 应用自然语言理解（NLP）

◎ 大作业6：自选

- 实验目的
  - 自定
- 实验要求
  - 书写实验报告，图文并茂。
- 注：须经老师认可