西北工业大学
NORTHWESTERN POLYTECHNICAL UNIVERSITY

C程序设计

Programming in C

1011014

主讲：姜学锋，计算机学院

函数之间实现批量数据传递

1、大数类型的接口设计

2、大数类型的实现

6.3.2 数组参数的传递机制

【大数运算和高精度运算】

计算整数

333333333333333333333333333333

+-×÷ 22222222222222222222222222222222

加、减、乘、除四则运算。

6.3.2 数组参数的传递机制

例题分析

C语言现有的数据类型

类型	类型标识符	内存长度	数值范围
整型	int	4	-2147483648～+2147483647
无符号整型	unsigned int	4	0～4294967295
长整型	long	4	-2147483648～+2147483647
无符号长整型	unsigned long	4	0～4294967295
长长整型	long long int __int64	8	9223372036854775807～ -9223372036854775808
无符号长长整型	unsigned long long int unsigned __int64	8	0～18446744073709551615

6.3.2 数组参数的传递机制

例题分析

显然，这是一个大数计算问题。

大数计算的运算对象和结果精度一般是少则数十位，多则几万位。在C语言内置数据类型中，精度最多只有二十多位，数值位数的长度是有限的。因此，大数计算不能直接用C语言的内置数据类型实现出来。

所以，编写程序时，需要首先设计出能够表示“大数”的数据类型，再设计算法求解。

6.3.2 数组参数的传递机制

例题分析

（1）由于需要表示无限位的整数，所以用整型数组来存放大数，定义为“大数类型”。例如：

```
int A[N]; //大数类型
```

需要特别规定：

A[0]存放大数最高位下标的偏移值（相对于A[2]）

A[1]存放大数的符号，为1表示正数，为-1表示负数

A[2]存放个位，A[3]存放十位，A[4]存放百位……，A[A[0]+2]存放最高位。

6.3.2 数组参数的传递机制

例题分析

（2）从“用户”角度来讲，如果输入输出
333333333333333333333333333333、

222222222222222222222222222222

的数据是直观的和方便的。但scanf、printf不支持像这样的大整数输入输出。为此，可以考虑使用字符数组。

按字符数组的方式调用scanf、printf函数（%s格式），并设计字符数组形式的“大数”转换为“大数类型”的函数str2big。

6.3.2 数组参数的传递机制

例题分析

（3）设计“大数类型”相关的转换函数

```
//将字符数组“大整数”s转换为“大数类型”A
void str2big(char s[], int A[]);
//将C语言整型n转换为“大数类型”A
void int2big(int n, int A[]);
```

6.3.2 数组参数的传递机制

例题分析

（4）设计“大数类型”的输入输出函数

```
void print(int A[]); //向控制台输出 “大数类型”
void scan(int A[]); //从控制台输入 “大数类型”
```

6.3.2 数组参数的传递机制

例题分析

（5）设计“大数类型”的实用函数

```
void assign(int A[],int B[]); //“大数类型”赋值 A=B
void zerojustify(int A[]); //调整大数A中高位无意义的0
int compare(int A[], int B[]);
//“大数类型”比较 A>B返回-1,A=B返回0,A<B返回1
void digitshift(int A[], int n); //“大数类型” A=A*10^n
```

6.3.2 数组参数的传递机制

例题分析

（6）设计“大数类型”的加、减、乘、除四则运算函数。

```
void Add(int A[], int B[], int R[]); //大数加法R=A+B
void Sub(int A[], int B[], int R[]); //大数减法R=A-B
void Mul(int A[], int B[], int R[]); //大数乘法R=AxB
void Div(int A[], int B[], int R[]); //大数除法R=A/B
```

6.3.2 数组参数的传递机制

例题分析

（7）Add(int A[], int B[], int R[])函数实现算法：

模拟小学生列竖式做加法，从个位开始逐位相加，超过或达到10则进位。

6.3.2 数组参数的传递机制

+ 加数	1	2	3	4	5	6	7	8	9	9
被加数	1	2	3	4	3	4	4	6	6	1

初始化进位为0，各对应位相加后再加上进位数

1、进位为1 [0]

2、进位为1 [6]

3、进位为1 [5]

4、进位为1 [2]

由低位向高位相加计算，直至所有运算结束

6.3.2 数组参数的传递机制

例题分析

（8）Sub(int A[], int B[], int R[])函数实现算法：

先分析A、B的各种情况：

①A为负：R=-A-B=-(A+B)

②B为负：R=A-(-B)=A+B

③B>A：R=-(B-A)

④逐位相减

6.3.2 数组参数的传递机制

例题分析

逐位相减算法：

算法从低位开始减，如果前一位相减有借位，就先减去上一位的借位，无则不减，再去判断是否能够减被减数，如果减不了，就要借位后再去减，同时置借位为1，否则置借位为0。

6.3.2 数组参数的传递机制

减数	1	2	3	4	5	6	7	8	6	7
− 被减数	1	2	3	4	3	4	5	7	9	8

初始化借位为0，各对应位相减后再减上借位数

1、借位为1 9

2、借位为1 6

3、借位为0 0

4、借位为0 2

由低位向高位相加计算，直至所有运算结束

6.3.2 数组参数的传递机制

例题分析

（9）Mul(int A[], int B[], int R[])函数实现算法：

①初始令row=A，R=0，由B的个位开始。

②执行r=r+row B次，即R=row*B的某位

③row=row*10

④处理B的高位

⑤重复执行②～⑤

6.3.2 数组参数的传递机制

例题分析

（10）Div(int A[], int B[], int R[])函数实现算法：

基本思想是反复做减法，看看从被除数里最多能减去多少个除数，商就是多少。

①先分别调整A和B的符号

②初始令row=A，tmp=0，R=0，按A逐位执行：

6.3.2 数组参数的传递机制

例题分析

（10）Div(int A[], int B[], int R[])函数实现算法：

③row=row*10，row位长为A某位值

④若row不小于B

⑤逐位相减 row-B，R对应位为相减数（即除法结果）

⑥重复执行④～⑥

⑦重复执行③～⑦

6.3.2 数组参数的传递机制

例6.60

```
 1 #include <stdio.h>
 2 #include <string.h>
 3 #define max(a,b) ((a)>(b)?(a):(b))
 4 #define N 102 //大数类型最大位数+2
 5 void Add(int A[], int B[], int R[]); //大数加法函数
 6 void Sub(int A[], int B[], int R[]); //大数减法函数
 7 void Mul(int A[], int B[], int R[]); //大数乘法函数
 8 void Div(int A[], int B[], int R[]); //大数除法函数
 9 //使用整型数组表示大数类型：其中A[0]=位数 A[1]=符号 A[2]...大数
10 void str2big(char s[], int A[]) //nnnnnn => A
11 { //字符串形式的“大数”转换为大数类型
12   int i;
13   for (i=0; i<N; i++) A[i]=0; //初始化
14   A[0] = -1; //初始时为NaN
15   for(i=strlen(s)-1; i>0; i--) {
```

6.3.2 数组参数的传递机制

例6.60

```
 1 #include <stdio.h>
 2 #include <string.h>
 3 #define max(a,b) ((a)>(b)?(a):(b))
 4 #define N 102 //大数类型最大位数+2
 5 void Add(int A[], int B[], int R[]); //大数加法函数
 6 void Sub(int A[], int B[], int R[]); //大数减法函数
 7 void Mul(int A[], int B[], int R[]); //大数乘法函数
 8 void Div(int A[], int B[], int R[]); //大数除法函数
 9 //使用整型数组表示大数类型：其中A[0]=位数 A[1]=符号 A[2]...大数
10 void str2big(char s[], int A[]) //nnnnnn => A
11 { //字符串形式的“大数”转换为大数类型
12   int i;
13   for (i=0; i<N; i++) A[i]=0; //初始化
14   A[0] = -1; //初始时为NaN
15   for(i=strlen(s)-1; i>0; i--) {
```

6.3.2 数组参数的传递机制

例6.60

```
 1 #include <stdio.h>
 2 #include <string.h>
 3 #define max(a,b) ((a)>(b)?(a):(b))
 4 #define N 102 //大数类型最大位数+2
 5 void Add(int A[], int B[], int R[]); //大数加法函数
 6 void Sub(int A[], int B[], int R[]); //大数减法函数
 7 void Mul(int A[], int B[], int R[]); //大数乘法函数
 8 void Div(int A[], int B[], int R[]); //大数除法函数
 9 //使用整型数组表示大数类型：其中A[0]=位数 A[1]=符号 A[2]...大数
10 void str2big(char s[], int A[]) //nnnnnn => A
11 { //字符串形式的“大数”转换为大数类型
12   int i;
13   for (i=0; i<N; i++) A[i]=0; //初始化
14   A[0] = -1; //初始时为NaN
15   for(i=strlen(s)-1; i>0; i--) {
```

6.3.2 数组参数的传递机制

例6.60

```
16      A[0]++; //记录最高位索引
17      A[ A[0]+2 ]= s[i]-'0'; //字符转换为数值
18    }
19    if( s[0]=='-' ) A[1]=-1; //符号为负
20    else { //否则为正数
21      A[0]++;
22      A[ A[0]+2 ]= s[0]-'0';
23      A[1] = 1; //符号为正
24    }
25  }
26  void int2big(int n, int A[])
27  { //整型转换为大数类型
28    int i, t;
29    for (i=0; i<N; i++) A[i]=0; //初始化
30    A[0] = -1; //初始时为NaN
```

6.3.2 数组参数的传递机制

例6.60

```
16      A[0]++; //记录最高位索引
17      A[ A[0]+2 ]= s[i]-'0'; //字符转换为数值
18    }
19    if( s[0]=='-' ) A[1]=-1; //符号为负
20    else { //否则为正数
21      A[0]++;
22      A[ A[0]+2 ]= s[0]-'0';
23      A[1] = 1; //符号为正
24    }
25  }
26  void int2big(int n, int A[])
27  { //整型转换为大数类型
28    int i, t;
29    for (i=0; i<N; i++) A[i]=0; //初始化
30    A[0] = -1; //初始时为NaN
```

6.3.2 数组参数的传递机制

例6.60

```
31    A[1] = n >= 0 ? 1 : -1; //由n确定符号
32    t = n>=0 ? n : -n;
33    while ( t>0 ) { //将n每1位设置到大数中
34      A[0]++;
35      A[ A[0]+2 ] = t % 10;
36      t = t / 10;
37    }
38    if (n==0) A[0]=0; //n为0，则大数也为0
39  }
40  void print(int A[])
41  { //输出大数
42    int i;
43    if ( A[1] == -1 ) printf("-"); //输出负号
44    for( i=A[0]; i>=0; i-- )
45      printf("%d", A[i+2]);
```

6.3.2 数组参数的传递机制

例6.60

```
31    A[1] = n >= 0 ? 1 : -1; //由n确定符号
32    t = n>=0 ? n : -n;
33    while ( t>0 ) { //将n每1位设置到大数中
34      A[0]++;
35      A[ A[0]+2 ] = t % 10;
36      t = t / 10;
37    }
38    if (n==0) A[0]=0; //n为0，则大数也为0
39  }
40  void print(int A[])
41  { //输出大数
42    int i;
43    if ( A[1] == -1 ) printf("-"); //输出负号
44    for( i=A[0]; i>=0; i-- )
45      printf("%d", A[i+2]);
```

6.3.2 数组参数的传递机制

例6.60

```
46    printf("\n");
47 }
48 void scan(int A[])
49 { //输入大数（字符串形式大整数）
50    char s[N-2];
51    scanf("%s",s); //输入大整数-nnnnnn
52    str2big(s, A);
53 }
54 void assign(int A[],int B[])
55 { //大数赋值A=B
56    int i;
57    for (i=0; i<N; i++) A[i]=B[i];
58 }
59 void zerojustify(int A[])
60 { //调整大数中无意义的0
```

6.3.2 数组参数的传递机制

例6.60

```
61    while( A[0] > 0 && A[ A[0]+2 ] == 0 )
62      A[0]--; //大数高位中的0无意义
63    if( A[0]== 0 && A[2] == 0 )
64      A[1] = 1; //避免出现 -0
65  }
66  int compare(int A[], int B[])
67  { //比较A和B -1(A>B) 0(A=B) 1(A<B)
68    int i;
69    if( A[1] == -1 && B[1] == 1) return 1; //A- < B+
70    if( A[1] == 1 && B[1] == -1) return -1; //A+ > B-
71    if( B[0] > A[0] ) return A[1]; //同号不同位数
72    if( A[0] > B[0] ) return -1*A[1]; //同号不同位数
73    for( i=A[0]; i >= 0; i-- ) { //同号同位数
74      if( A[ i+2 ] > B[ i+2 ] ) return -1*A[1];
75      if( B[ i+2 ] > A[ i+2 ] ) return A[1];
```

6.3.2 数组参数的传递机制

例6.60

```
76    }
77    return 0; //相等
78 }
79 void digitshift(int A[], int n)
80 { //计算A*10^n
81    int i;
82    if( A[0] == 0 && A[2] == 0 ) return; //为0
83    for( i=A[0]; i>=0; i--)
84      A[ i + n +2 ] = A[ i+2 ]; //大数向左移动n位
85    for( i=0; i<n; i++ )
86      A[ i+2 ] = 0; //低n位为0
87    A[0] = A[0] + n; //大数位长增加n
88 }
89 void Add(int A[], int B[], int R[])
90 { //大数加法R=A+B
```

6.3.2 数组参数的传递机制

例6.60

```
91    int i, c=0; //c为进位
92    int2big(0, R); //R=0
93    if( A[1] == B[1] ) R[1]=A[1]; //A、B同号
94    else {
95      if( A[1]==-1 ) { //A-则R=B-A
96        A[1] = 1;
97        Sub(B, A, R);
98        A[1] = -1;
99      }
100     else { //B-则R=A-B
101       B[1] = 1;
102       Sub(A, B, R);
103       B[1] = -1;
104     }
105     return;
```

6.3.2 数组参数的传递机制

例6.60

```
106     }
107     R[0] = max(A[0], B[0]) + 1; //和的位长
108     for(i=0; i<=R[0]; i++) { //逐位相加，考虑进位
109       R[ i+2 ] = (c + A[ i+2 ] + B[ i+2 ]) % 10;
110       c = (c + A[ i+2 ] + B[ i+2 ]) / 10;
111     }
112     zerojustify(R);
113 }
114 void Sub(int A[], int B[], int R[])
115 { //大数减法R=A-B
116     int i, v, b=0; //b为借位
117     int2big(0, R); //R=0
118     if( A[1] == -1 || B[1] == -1 ) { //R=A- -B,R=-A-B做加法
119       B[1] = -1 * B[1];
120       Add(A, B, R);
```

6.3.2 数组参数的传递机制

例6.60

```
106     }
107     R[0] = max(A[0], B[0]) + 1; //和的位长
108     for(i=0; i<=R[0]; i++) { //逐位相加，考虑进位
109       R[ i+2 ] = (c + A[ i+2 ] + B[ i+2 ]) % 10;
110       c = (c + A[ i+2 ] + B[ i+2 ]) / 10;
111     }
112     zerojustify(R);
113 }
114 void Sub(int A[], int B[], int R[])
115 { //大数减法R=A-B
116     int i, v, b=0; //b为借位
117     int2big(0, R); //R=0
118     if( A[1] == -1 || B[1] == -1 ) { //R=A- -B,R=-A-B做加法
119       B[1] = -1 * B[1];
120       Add(A, B, R);
```

6.3.2 数组参数的传递机制

例6.60

```
121       B[1] = -1 * B[1];
122       return;
123     }
124     if(compare(A, B) == 1 ) { //B>A则R=-(B-A)
125       Sub(B, A, R);
126       R[1] = -1;
127       return;
128     }
129     R[0] = max(A[0], B[0]); //减的位长
130     for(i=0; i<=R[0]; i++) {
131       v = A[ i+2 ] - b - B[ i+2 ]; //逐位相减
132       if( A[ i+2 ] > 0) b = 0;
133       if( v < 0 ) { //处理借位
134         v = v + 10;
135         b = 1;
```

6.3.2 数组参数的传递机制

例6.60

```
136       }
137       R[i+2] = v % 10;
138     }
139     zerojustify(R);
140 }
141 void Mul(int A[], int B[], int R[])
142 { //大数乘法R=AxB
143     int tmp[N], row[N];
144     int i, j;
145     int2big(0, R); //R=0
146     assign(row,A); //row=A
147     for(i=0; i<=B[0]; i++ ) { //B逐位乘A
148       for(j=1; j<=B[i+2]; j++) { //多次相加
149         Add(R, row, tmp); //R=R+row*B
150         assign(R,tmp);
```

6.3.2 数组参数的传递机制

例6.60

```
136       }
137       R[i+2] = v % 10;
138     }
139     zerojustify(R);
140 }
141 void Mul(int A[], int B[], int R[])
142 { //大数乘法R=AxB
143     int tmp[N], row[N];
144     int i, j;
145     int2big(0, R); //R=0
146     assign(row,A); //row=A
147     for(i=0; i<=B[0]; i++ ) { //B逐位乘A
148       for(j=1; j<=B[i+2]; j++) { //多次相加
149         Add(R, row, tmp); //R=R+row*B
150         assign(R,tmp);
```

6.3.2 数组参数的传递机制

例6.60

```
151       }
152       digitshift(row, 1); //下次row=row*10
153     }
154     R[1] = A[1] * B[1]; //处理相乘时的符号
155     zerojustify(R);
156 }
157 void Div(int A[], int B[], int R[])
158 { //大数除法R=A/B
159     int tmp[N], row[N];
160     int i, asign, bsign;
161     int2big(0, R); //R=0
162     R[1]=A[1]*B[1]; //处理相除时的符号
163     asign = A[1]; //保存A的符号
164     bsign = B[1]; //保存B的符号
165     A[1] = 1; //A符号调整为正
```

6.3.2 数组参数的传递机制

例6.60

```
151       }
152       digitshift(row, 1); //下次row=row*10
153     }
154     R[1] = A[1] * B[1]; //处理相乘时的符号
155     zerojustify(R);
156   }
157   void Div(int A[], int B[], int R[])
158   { //大数除法R=A/B
159     int tmp[N], row[N];
160     int i, asign, bsign;
161     int2big(0, R); //R=0
162     R[1]=A[1]*B[1]; //处理相除时的符号
163     asign = A[1]; //保存A的符号
164     bsign = B[1]; //保存B的符号
165     A[1] = 1; //A符号调整为正
```

6.3.2 数组参数的传递机制

例6.60

```
166    B[1] = 1; //B符号调整为正
167    int2big(0, row); //row=0
168    int2big(0, tmp); //tmp=0
169    R[0]=A[0]; //R的位长初始与A相同
170    for(i=A[0]; i >= 0; i--) {
171      digitshift(row, 1); //row=row*10
172      row[ 2 ] = A[ i+2 ];
173      R[ i+2 ] = 0;
174      while ( compare(row, B) != 1 ) { //row<B
175        R[ i+2 ]++; //结果为相减次数
176        Sub(row, B, tmp); //逐位相减 row-B
177        assign(row, tmp);
178      }
179    }
180    zerojustify(R);
```

6.3.2 数组参数的传递机制

例6.60

```
181     A[1] = asign; //A符号还原
182     B[1] = bsign; //B符号还原
183 }
184 int main()
185 { //调用大数运算函数计算结果
186     int c, A[N], B[N], R[N], Z[N];
187     printf("a=");
188     scan(A); //输入大数A
189     printf("b=" );
190     scan(B); //输入大数B
191     Add(A, B, R); //计算大数加法R=A+B
192     printf("a+b=");
193     print(R);
194     c = compare(A, B); //大数比较
195     printf("a %s b\n", c==0 ? "==" : ( c<0 ? ">" : "<" ));
```

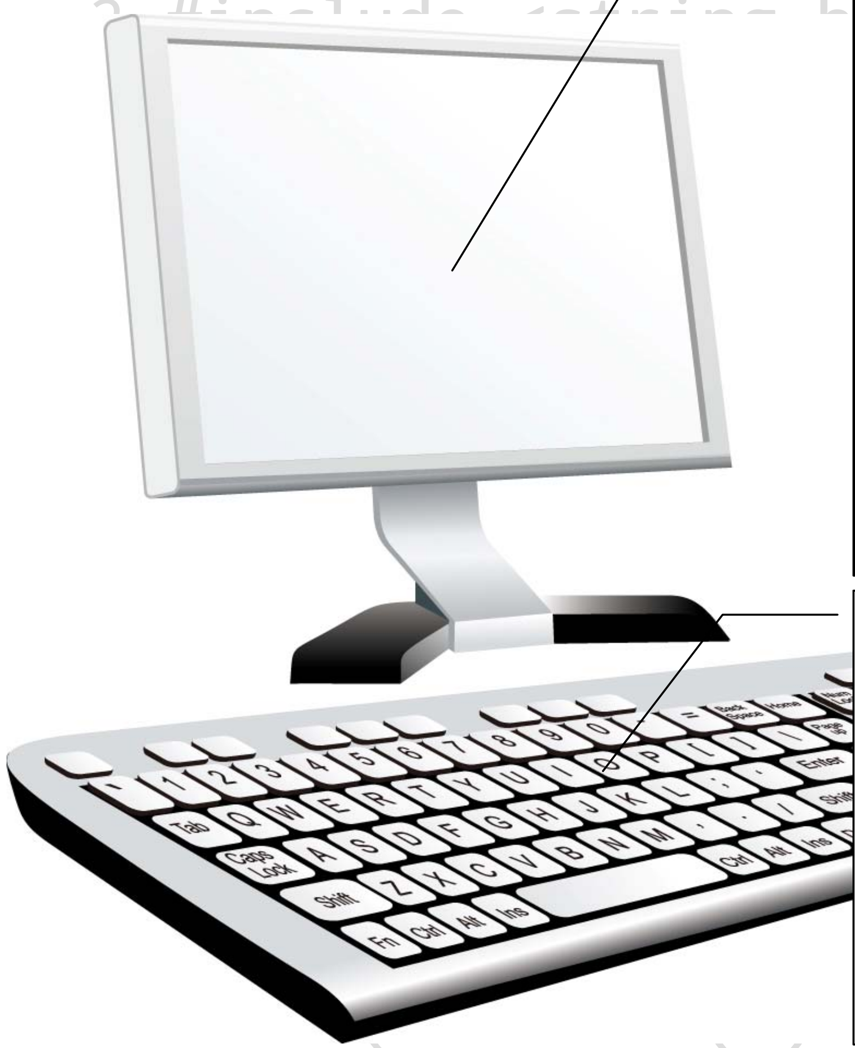
6.3.2 数组参数的传递机制

例6.60

```
196     Sub(A, B, R); //计算大数减法R=A-B
197     printf("a-b=");
198     print(R);
199     Mul(A, B, R); //计算大数乘法R=AxB
200     printf("a*b=");
201     print(R);
202     int2big(0, Z);
203     if(compare(Z, B)==0) printf("a/b=NaN\n"); //不能除0
204     else {
205       Div(A, B, R); //计算大数除法R=A/B
206       printf("a/b=");
207       print(R);
208     }
209     return 0;
210 }
```

6.3.2 数组参数的传递机制

例6.60



```
a+b=5555555555555555555555555555555555555555
a > b
a-b=1111111111111111111111111111111111111111
a*b=7407407407407407407407407407407407407405925925
925925925925925925925925925925926
a/b=1
```

```
3333333333333333333333333333333333333333↙
2222222222222222222222222222222222222222↙
```


程序设计