**电子科技大学信息与软件工程学院**

**标 准 实 验 报 告**

**（实验）课程名称 信息安全数学基础实验**

**电子科技大学教务处制表**

**电 子 科 技 大 学**

**实 验 报 告**

**学生姓名：周玉川 学 号：2017221302006 指导教师： 陈大江**

**实验地点：3教410 实验时间：2018.09.22**

**一、实验室名称： 网络安全实验室**

**二、实验项目名称：多精度数运算的实现**

**三、实验学时：2学时**

**四、实验原理：**

大数指的是超过计算机CPU寄存器表达的数，即超过计算机字长的数。大数基本运算主要指的是对大数进行数论运算，如加、减、乘、除。出于效率原因，一般的大数运算主要指对无符号类型的数进行数论计算。

目前主流RSA算法都建立在512位到1024位的大数运算之上。然而大多数的编译器只能支持到64位的整数运算，即运算中整数必须小于等于64位，即：0xFFFFFFFFFFFFFFFF，这远远达不到RSA的需要，于是需要建立专门的大数运算库来解决这一问题。

最简单的办法是将大数当作字符串处理，也就是将大数用10进制字符数组进行表示，然后模拟人们手工进行“竖式计算”的过程编写其加减乘除函数。但是这样做效率很低，因为1024位的大数其10进制数字个数就有数百个，对于任何一种运算，都需要在两个有数百个元素的数组空间上做多重循环，还需要许多额外的空间存放计算的进位退位标志及中间结果。其优点是算法符合人们的日常习惯，易于理解。

另一种方法是将大数当作一个二进制流进行处理，使用各种移位和逻辑操作来进行加减乘除运算，但是这样做代码设计非常复杂，可读性很低，难以理解也难以调试。

**五、实验目的：**

1. 掌握一种1024位大数运算原理；
2. 依据所选择的算法，编程实现该算法；

**六、实验内容：**

本实验要求学生掌握常用的大数运算算法的实现方法，并运用高级程序设计语言完成一种大数运算算法的程序，加深对大数运算的理解。

**七、实验步骤：**

类及其成员函数、变量接口说明如表1.。

表1.4类及其成员函数、变量接口说明

|  |  |
| --- | --- |
| 类名 | BigNum |
|  | |
| 成员函数 | public int bigPlus (int a[],int b[],int c[]) |
| 名称 | 大数加法运算 |
| 接口 | int bigPlus (int a[],int b[],int c[]) |
| 输入 | int a[],int b[] |
| 输出 | int c[] |
| 功能 | 计算大数a+b |
| 出错 | 返回-1 |
|  | |
| 成员函数 | public int bigSub (int a[],int b[],int c[]) |
| 名称 | 大数减算 |
| 接口 | int bigSub (int a[],int b[],int c[]) |
| 输入 | int a[],int b[] |
| 输出 | int c[] |
| 功能 | 计算a-b |
| 出错 | 返回-1 |
|  | |
| 成员函数 | public int bigMult (int a[],int b[],int c[]) |
| 名称 | 大数乘算 |
| 接口 | int bigMult (int a[],int b[],int c[]) |
| 输入 | int a[] ，int b[] |
| 输出 | int c |
| 功能 | 计算a\*b |
| 出错 | 返回-1 |

程序代码如下：

|  |
| --- |
| 加法:  void bigPlus ( int a[], int b[], int c[] ){  int max\_len = ( a[0] > b[0] ? a[0] : b[0] );  int i;    for ( i = 0 ; i < max\_len+2 ; i++){  c[i]=0;  }  int carry=0;//进位初始化为0    for ( i = 1 ; i < max\_len+1 ; i++){  c[i]=a[i]+b[i]+carry;  carry=c[i]/10;  c[i]=c[i]%10;  }  if (carry){  c[i]=1;  max\_len++;// 若还有进位，位数加一  }  c[0]=max\_len;  } |
| 减法:  void bigSub ( int a[], int b[], int c[]){//默认a可以减b a[0]>=b[0]  int i;  int h=0;//借位初始化为0  for ( i = 0 ; i < a[0]+1 ; i++){  c[i]=0;  }  for ( i = 1 ; i < b[0]+1 ; i++){  c[i]=a[i]-b[i]+h;  if (c[i] < 0){  c[i]+=10;  h=-1;  }  else{  h=0;  }  }  for ( ; i < a[0]+1 ; i++ )  {  c[i]=a[i]+h;  if (c[i] < 0)  {  c[i]+=10;  h=-1;  }  else{  h=0;  }  }  for ( i = a[0] ; i > 0 ; i--)  {  if (c[i] > 0 ){  break;  }  }  c[0]=i;  } |
| 乘法:  void bigMult ( int a[], int b[], int c[] ){  int i,j;  for ( i = 0 ; i < a[0]+b[0]+2 ; i++){  c[i]=0;  }  for ( i = 1 ; i < a[0]+1 ; i++){  for ( j = 1 ; j < b[0]+1 ; j++){  c[i+j-1]+=a[i]\*b[j];  }  }  for ( i = 1 ; i < a[0]+b[0] ; i++){  if ( c[i] >= 10){  c[i+1]+=c[i]/10;  c[i]%=10;  }  }  //a和b积的最大长度为。a[0]+b[0],最小为a[0]+b[0]-1  if (c[i]==0){ //c[i]长度为第二种情况  i--;  }  c[0]=i;  } |
| Main函数：  #include <bits/stdc++.h>  #define MAX\_LEN 100  using namespace std;  void bigPlus ( int a[], int b[], int c[] );  void bigSub ( int a[], int b[], int c[]);  void bigMult ( int a[], int b[], int c[]);  void print(int a[]);  int main(void){  int a[MAX\_LEN],b[MAX\_LEN],c[2\*MAX\_LEN];  char str[MAX\_LEN];  int i;  int len;  memset(a,0,sizeof(a));//初始化为0  memset(b,0,sizeof(b));  memset(c,0,sizeof(c));    gets(str);  len=strlen(str);//读取，反转  for ( i = 1 ; i < len+1 ; i++){  a[i]=str[len-i]-'0';  }  a[0]=len;    gets(str);  len=strlen(str);//读取b，反转  for ( i = 1 ; i < len+1 ; i++){  b[i]=str[len-i]-'0';  }  b[0]=len;  printf("\na: ");print(a);//输出a，b  printf("b: ");print(b);    printf("\na+b:\n");//a+b  bigPlus(a,b,c);  print(c);    printf("\na-b:\n");//a-b  bigSub(a,b,c);  print(c);    printf("\na\*b: \n");//a\*b  bigMult(a,b,c);  print(c);    return 0;//结束  } |

**九、实验数据及结果分析：**

实现范例测试界面如图1.1，测试界面代码请参考附录。该示范程序用VC++高级语言实现。使用时依次在a,b后面输入数值，点击加，减，乘计算按钮，计算结果在其后面显示出来。

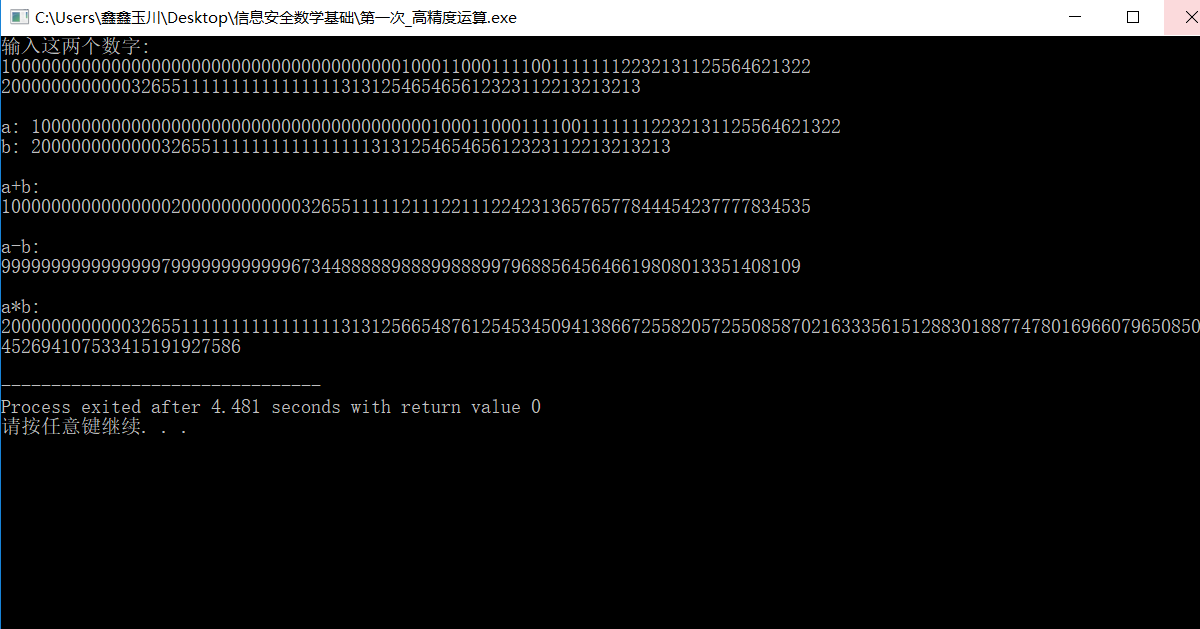


图1.1大数实现范例

图1.1 为两个64bit以上长度的数进行加法，减法，乘法运算所得结果。用软进行验证，结果正确。

**十、实验结论：**

采用数组的方法，结合数组操作，成功实现大整数的加、减、乘运算，实验成功。

**十一、总结及心得体会：**

大数运算主要是将大数转换为b进制表示，利用数组进行存储再进行处理。

**十二、对本实验过程及方法、手段的改进建议：**

1. 浪费空间：实验中一个整型变量只存一位（0-9），建议存4位及以上。
2. 浪费时间：每次加减只处理一位，建议一次处理多位，以节约时间。
3. 可以鼓励用c语言之外的语言实现功能，避免学生只依赖于以前有限的知识，促进自主学习。

**报告评分：**

**指导教师签字：**