字符串转换成整数

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<math.h>

#include<limits.h>

int flag = 1;//全局变量，作为函数的返回值

int FormIsOk(char\*);//检测字符串格式是否正确

bool Overflow(long, long, long);//判断两个整数相加是否溢出

bool tmult\_ok(long, long, long);//判断两个整数相乘是否溢出

void EveryCharToInt(char\*);//字符串各bit的char转换成int

bool PositiveStringToInt(char\*, long\*);//表示正数的字符串转换成整数

long int StringToInt(char\*);//任意合法字符串转换成整数

/\*

功能:字符串转换成整数.

flag = 1：成功转换并返回了正确结果

flag = -2且返回值是-2 :字符串格式错误

flag = 2且返回值是2：字符串是空

flag = 3：大数溢出，无法转换成整型

\*/

long int StringToInt(char \*string) {

long int sum;

int result = FormIsOk(string);

if (result == -2) {

flag = -2; return 0;/\*(可以返回任意数)字符串格式错误\*/

}

/\*开始转换，字符串是0时，返回0\*/

else if (result == 0) {

flag = 1; return 0;

}

else if (result == -1){

char \*p = string + 1;

bool b = PositiveStringToInt(p, &sum);

if (b) flag = 1;

else flag = 3;

return -sum;

}

else {

bool b = PositiveStringToInt(string, &sum);

if (b) flag = 1;

else flag = 3;

return sum;

}

}

/\*

检测一个字符串的格式是否正确.

flag是-2，格式错误或空字符串

flag是-1，格式正确且字符串表示负数

flag是0，格式正确且字符串表示0

flag是1，格式正确且字符串表示正数

\*/

int FormIsOk(char \*string) {

/\*\*/

char \*p = string;

if (p == NULL)

return -2;/\*空字符串返回- 2\*/

if (\*p == '-') {

++p;

while (\*p != '\0' && \*p == '0') {

++p;

}

if (\*p == '\0' || \*p != \*(string + 1))

return -2;

while (\*p != '\0') {

if (!(\*p >= '0' && \*p <= '9'))

return -2;

++p;

}

return -1;/\*负数返回-1\*/

}

else if (\*p == '0') {

if (\*++p == '\0')

return 0;/\*0返回0\*/

return -2;

}

else {

while (\*p != '\0') {

if (!(\*p >= '0' && \*p <= '9'))

return -2;

++p;

}

return 1;/\*正数返回1\*/

}

}

/\*判断两个数相加是否溢出,溢出返回0,没溢出返回1\*/

bool Overflow(long x, long y, long sum) {

int flag1 = x > 0 && y > 0 && sum <= 0;

int flag2 = x < 0 && y < 0 && sum >= 0;

return (!flag1) && (!flag2);

}

/\*判断两个整数相乘是否溢出\*/

bool tmult\_ok(long x, long y, long p){

return !x || p / x == y;

}

/\*把一个非空字符串每位由char型转换成int类型\*/

void EveryCharToInt(char \*str) {

while (\*str != '\0') {

\*str -= '0';

++str;

}

}

/\*

返回一个表示正数的字符串转换成的整型值

溢出返回false,没溢出返回true

\*/

bool PositiveStringToInt(char \*str, long \*finalSum) {

EveryCharToInt(str);

long sum = 0, weight = 1, beforeSum = 0, beforeWeight = 1;

int length = strlen(str);

for (int i = length - 1; i >= 0; --i) {

sum += str[i] \* weight;

if (!Overflow(beforeSum, str[i] \* weight, sum)) {

return false;

}

beforeSum = sum;

weight \*= 10;

if (!tmult\_ok(beforeWeight, 10, weight)) {

return false;

}

beforeWeight = weight;

}

\*finalSum = sum;

}

int main() {

char p[100] = "2147438648";

long long int num;

//123456789

num = StringToInt(p);

if (flag == 3)

printf("溢出\n");

else if (flag == 1)

printf("num = %lld\n", num);

else if (flag == 2)

printf("空字符串\n");

else

printf("字符串格式错误\n");

system("pause");

return 0;

}

字符串表示大数的原理

一个大数【348992932655663433878783783763577634672902013880134778372599】

如何表示？

（1）int数组，数组容量不小于大整数的位数，每个单元存放对应位的字面值，多出来的高 位填0，以%d输出。

（2）char整型数组，每个单元存放的是整形数据，和（1）中的int数组方法完全一致，以% d输出。

（3）char字符数组，每个单元存放相应字面值的象形字符（非相等整型字面值，但通过减 去一个字面值即可得到实际字面值），多出来的高位填‘0’，以%c输出。

如何运算？

（1）（2）遵从人类计数满进制进位规则。

（3）比较特殊。本应存放实际字面值，但实际放置的是非相等字面值编码表示的实际字面值对应的字符。运算时，先把非相等字面值通过减去一个特定字面值得到实际字面值，再运算，此时的运算编程（1）（2）中的运算。运算完毕再通过加一个特定字面值恢复实际字面值对应的表示字符的非相等字面值。运算时，需要加减的整数，既可以用字面值，又可以用字符（字符完全等价于整数，类比于#define 字符 整数）.

大数运算

/\*

实现功能：

1.字符串表示大数

2.字符串表示的大数加1操作

3.两个字符串表示的大数的加法运算

4.打印一个字符串表示的大数

\*/

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

/\*MAXSTRSIZE表示本程序所有大数所占字节数\*/

#define MAXSTRSIZE 100

/\*

字节数不是标准字节数MAXSTRSIZE转换成标准字节数目表示的大数；

类比于int型转换成long long int型。

返回转换好的字符大数的基地址。

\*/

char\* Translen(char \*str) {

int i;

int len = strlen(str);/\*获取待转换字符串大数的长度\*/

char\* p = (char\*)malloc(sizeof(char)\*(MAXSTRSIZE + 1));

if (!p) exit(-1);

p[MAXSTRSIZE] = '\0';

for (i = len - 1; i >= 0; --i)

p[MAXSTRSIZE - len + i] = str[i];

for (i = MAXSTRSIZE - len - 1; i >= 0; --i)

p[i] = '0';/\*多余的高位单元赋'0'\*/

return p;

}

/\*以字符形式输出用字符串表示的大数\*/

/\*适用一切进制\*/

void printf\_char(char \*str) {

int i = 0;

while (i <= MAXSTRSIZE - 1 && str[i] == '0')

++i;/\*去掉高位的'0'\*/

if (i == MAXSTRSIZE)

printf("0");

else/\*从非'0'的最高位开始打印\*/

while (i <= MAXSTRSIZE - 1)

printf("%c", str[i++]);

printf("\n");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*表示大数的字符串转换为整型数组表示的大数\*/

/\* A10,B11,C12,D13,E14,F15 \*/

void str\_to\_int(char \*str) {

while (\*str != '\0') {

if (\*str >= '0'&&\*str <= '9')

\*str -= '0';

if (\*str >= 'A'&&\*str <= 'F')

\*str -= '7';

++str;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*字符串表示的十进制大数实现加1操作\*/

bool AddByOne(char \*str) {

/\*所有权位执行加法时，操作是相同的\*/

int flag = 0;/\*发生进位继续加，没发生停止运算\*/

int i;

for (i = MAXSTRSIZE - 1; i >= 0; --i) {

/\*操作所有位\*/

str[i] = str[i] - '0' + 1;/\*转换成实际字面值进行运算\*/

/\*加法肯定加在最低位\*/

if (str[i] == 10) {

/\*满足进位条件，本位清零，并进位1\*/

str[i] = 0 + '0';/\*实际字面值转换成编码对应的非字面值\*/

flag = 1;

/\*标志位置1，发生进位，继续操作下一个高位\*/

}

else {

str[i] += '0';/\*别tm忘了加\*/

flag = 0;

}

if (flag == 0)

break;

/\*没发生进位，标志位置0，终止运算\*/

}

/\*最高位发生进位，溢出，加法失效，视作异常情况\*/

if (i == -1)

return 0;

/\*没发生最高位进位，加法有效\*/

return 1;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*字符串表示的十进制大数加法进位\*/

char\* AddTwoNum(char \*str1, char \*str2) {

char \*str = (char\*)malloc(sizeof(char)\*(MAXSTRSIZE + 1));

if (!str) exit(-1);

str[MAXSTRSIZE] = '\0';

for (int k = 0; k <= MAXSTRSIZE - 1; ++k)

str[k] = '0';

int i = MAXSTRSIZE - 1;

char isum;

while (i>0) {/\*要防止数组越界\*/

isum = str1[i] - '0' + str2[i] - '0';

if (isum >= 10) {

/\*此处进位，进到被加数上，而不是存放计算结果的单元上\*/

++str1[i - 1];/\*tmd，在这犯了错误！！我真是傻逼\*/

str[i] = isum - 10;/\*去掉进位后的实际字面值\*/

str[i] += '0';/\*整型再转换成字符型\*/

}

else

str[i] = isum + '0';

--i;

}

/\*为防止数组越界而单独处理a[0]单元\*/

isum = str1[0] - '0' + str2[0] - '0';

if (isum >= 10)

return NULL;

else {

str[0] = isum + '0';

return str;

}

}

int main() {

char str1[20] = "123456789";

char str2[20] = "1673211";

char \*str11 = Translen(str1);

char \*str22 = Translen(str2);

/\*检验打印函数\*/

printf("打印第一个数组str11:\n");

printf\_char(str11);

printf("打印第二个数组str22:\n");

printf\_char(str22);

/\*检测加1函数\*/

printf("第一个字符串加1后的结果:\n");

if (AddByOne(str11))

printf\_char(str11);

else

printf("溢出，加法失败！\n");

printf("第一个字符串加1后的结果:\n");

if (AddByOne(str22))

printf\_char(str22);

else

printf("溢出，加法失败！\n");

/\*检测两大数加法运算\*/

printf("两个数相加后的结果:\n");

char \*suntwostr = AddTwoNum(str11, str22);

if (suntwostr)

printf\_char(suntwostr);

else

printf("溢出，加法失败！\n");

return 0;

}

打印1 to n位最大整数

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

char\* strsuper(int n) {

char \*str = (char\*)malloc(sizeof(char)\*(n + 1));

if (!str) exit(-1);

int i;

for (i = 0; i<n - 1; ++i) {

str[i] = '0';

}

str[n - 1] = '1';

str[n] = '\0';

return str;

}

/\*字符串表示的大数加1\*/

bool strincrement(char \*str) {

int len = strlen(str);

int flag = 0;

int i;

for (i = len - 1; i >= 0; --i) {

str[i] += 1;

if (str[i] == ':') {

str[i] = '0';

flag = 1;

}

else

flag = 0;

if (flag == 0)

break;

}

if (i == -1)

return 0;

return 1;

}

void printfint(char\* number) {

int len = strlen(number);

int i = 0;

while (i <= len - 1 && number[i] == '0') {

i++;

}

while (i <= len - 1) {

printf("%c", number[i]);

i++;

}

printf("\t");

}

int main() {

char \*str = strsuper(2);

printfint(str);

while (strincrement(str)) {

printfint(str);

}

return 0;

}

进制转换

/\*功能：任意m进制的正整数转换成任意n进制的正整数，2<=m≠n<=16 \*/

/\*流程：1，先把m进制转化成十进制

2，再把十进制转化成n进制（我使用了递归，其实可不用递归）

\*/

#include<stdio.h>

#include<string.h>

typedef int Status;//返回状态，相当于bool

#define ok 1

#define error 0

/\*k变量是全局变量，极其重要！！！

重要的话只说一遍！！！

k被实现十进制转换成n进制的函数引入，用于把每一次的余数，

放进数组，k是索引，每次都要k++；但是，递归一次就要初始化一次k，

所以k不能放进函数里面。故把k设置成全局变量，实现只初始化一次的目的。

另外k还有一个作用：k的值就是结果数组元素的数目，Trans\_m\_To\_n函数输出

结果时，能很轻松的获知输出几位。

【注释】，其实k也可设置成TenToTrans的局部变量，同时，需要函数多个形参，

用来把k的值传出去。

\*/

int k = 0;/\*\*\*相当重要的一行代码\*\*\*/

/\*实现m进制转换成十进制\*/

Status TransToTen(char \*str, int m, unsigned long &num) {

int i = strlen(str);/\*因为16进制有字符，超过9，所以要以字符形式输入

被转化数，而不是int\*/

int j = i - 1;

num = 0;

unsigned long temp;

if (!(0 <= m&&m <= 16))//功能说了解决2--16进制的转换

return error;

while (j >= 0) {

if ('0' <= str[j] && str[j] <= '9')

str[j] -= '0';

else if ('a' <= str[j] && str[j] <= 'f')

str[j] = str[j] - '0' - 39;

else if ('A' <= str[j] && str[j] <= 'F')

str[j] = str[j] - '0' - 7;

else

return error;

if (str[j] >= m)//m进制的数字肯定小于m

return error;

if (i - 1 == j)//如果用math.h函数库，多次计算指数耗时

temp = 1;//此代码采用权位累积，避免了繁琐而无谓的运算

else

temp \*= m;

num += str[j] \* temp;//累加器

--j;//循环控制

}

return ok;

}

/\*任意十进制转换为n进制\*/

Status TenToTrans(unsigned int num, int n, int \*a) {

if (!(2 <= n && n <= 16))

return error;

if (0 == num)

return error;

a[k++] = num % n;

num = (unsigned long)num / n;

TenToTrans(num, n, a);

return ok;

}

/\*实现m进制直接转换成n进制\*/

void Trans\_m\_To\_n(int m, int n, char \*str) {

int t;

int a[100];

unsigned long num;

TransToTen(str, m, num);

TenToTrans(num, n, a);

t = k - 1;

while (t >= 0)

printf("%d ", a[t--]);

}