深蓝学院
shenlanxueyuan.com

大数加法及进制转换代码讲解

主讲人 云行月下

纲要

- 第一部分：整体思路
- 第二部分：输入合规检查
- 第三部分：大数类设计与构建
- 第四部分：大数加法
- 第五部分：进制转换

整体思路

- 基本思路为设计一个数值类并实现加法和取余
- 基本流程为：

  判断输入合规 ↓

  构造数值类 ↓

  计算加法（减法也作为加法处理） ↓

  转换到目标进制 ↓

  输出结果

```
std::string num1;
std::string num2;

std::getline(infile, num1);
std::getline(infile, num2);

infile.close();

std::cout << "Input is :" << std::endl;
Num&& n1 = makeNum(num1, format);
Num&& n2 = makeNum(num2, format);

Num&& result = n1 + n2;

if (format != display_format)
    result = reFormat(result, display_format);

std::cout << "Result is :" << std::endl;
dispNum(result);
```

纲要

深蓝学院
shenlanxueyuan.com

- 第一部分：整体思路
- **第二部分：输入合规检查**
- 第三部分：大数类设计与构建
- 第四部分：大数加法
- 第五部分：进制转换

输入合规检查

- 目标效果

```
@noetic:~/Documents$ ./awesome_format unknowfile
Open file unknowfile failed!
@noetic:~/Documents$ ./awesome_format inputfile 1
Format parameter input out of range!
@noetic:~/Documents$ ./awesome_format inputfile nonenum
Format parameter is not a number!
@noetic:~/Documents$ ./awesome_format
File name should be entered!
@noetic:~/Documents$ ./awesome_format inputfile 10 10 unexpect_param1 unexpect_param2
Cannot parse the parameter:
unexpect_param1
unexpect_param2
@noetic:~/Documents$ ./awesome_format inputfile 9
Input is :
123143523425239072
--------------^
Number format error!
Aborted (core dumped)
@noetic:~/Documents$ ./awesome_format inputfile
Input is :
123143523425239072
53214324967686897563
Result is :
53337468491112136635
```

输入合规检查

- 实现代码，需要注意switch中的fallthrough

```
char format = 10;
char display_format = 10;
try
{
    switch (argc)
    {
    case 4: display_format = std::stoi(argv[3]);
    case 3: format = std::stoi(argv[2]);
    case 2: break;
    default:
        if (argc < 2)
            std::cerr << "File name should be entered!" << std::endl;
        else
            std::cerr << "Cannot parse the parameter:" << std::endl;
        for (size_t i = 4; i < argc; i++)
            std::cerr << argv[i] << std::endl;
        return -1;
        break;
    }
}
```

```
catch (const std::invalid_argument&)
{
    std::cerr << "Format parameter is not a number!" << std::endl;
    return -1;
}

if (format < 2 || format>36 || display_format < 2 || display_format>36)
{
    std::cerr << "Format parameter input out of range!" << std::endl;;
    return -1;
}

std::ifstream infile(argv[1]);
if (!infile.is_open())
{
    std::cerr << "Open file " << argv[1] << " failed!" << std::endl;
    return -1;
}
```

纲要

深蓝学院
shenlanxueyuan.com

- 第一部分：整体思路
- 第二部分：输入合规检查
- **第三部分：大数类设计与构建**
- 第四部分：大数加法
- 第五部分：进制转换

大数类设计与构建

- 使用数组来表示大数，高位在前，使用std::vector或其他容器也可以实现相同的效果
- 使用真实值保存大数的每一位
- 大数的符号和进制单独保存
- 需要提供访问每一位的方法
- 需要设计加法和取余的符号重载函数
- 为了进制转换的方便，需要设计判断当前大数是否为0的方法

大数类设计与构建

- 大数类结构框架
- 大数加法在类外实现

```
//核心类型，负责储存大数，管理内存
struct Num
{
    Num(size_t length, char format, bool positive) : v(new size_t[length]), length(length), format(format), positive(positive) {}

    //提供类似数组的访问方法
    inline size_t& operator[](size_t i) { return v[i]; }

    //提供类似数组的访问方法，但是取得的是ascii，通过将size_t解释为NumToken实现
    inline char operator()(size_t i) { return reinterpret_cast<NumToken*>(v)[i].toASCII(); }

    ~Num() { if (v != nullptr) delete[] v; }

    //防止不期望的拷贝
    Num(const Num&) = delete;

    //移动构造，实现基础移动语义，使语法更加优雅
    Num(Num&& old) noexcept : v(old.v), length(old.length), format(old.format), positive(old.positive) { old.v = nullptr; }

    //移动拷贝，实现基础移动语义，使语法更加优雅
    Num& operator=(Num&& old) { ... }

    //求余，进制转换输出核心，需要注意的是，求余也会将自身变为商，和正常的求余略有不同
    unsigned char operator%(unsigned char f) { ... }

    //判断当前大数是否为0，用于进制转换输出结束条件
    bool isZero() { ... }

    //删除高位0
    void autoResize() { ... }

    bool positive;
    char format;
    size_t* v;
    size_t length;
};
```

大数类设计与构建

- 从ascii到真实值的转换可以设计辅助类来完成，代码如下
- 由于辅助类仅有一个size_t成员，所以他和size_t具有相同的布局，我们可以通过reinterpter_cast将NumToken直接转换成size_t

```
//用于解释输入流，可以看成是size_t类型的另一种解释
struct NumToken
{
    //将ASCII转为真实值
    NumToken(char ascii)
    {
        if (ascii >= '0' && ascii <= '9')
            v = ascii - '0';
        else if (ascii >= 'a' && ascii <= 'z')
            v = ascii - 'a' + 10;
        else if (ascii >= 'A' && ascii <= 'Z')
            v = ascii - 'A' + 10;
        else
            v = 256;
    }
```

```
    //输出显示
    char toASCII()
    {
        if (v < 10)
            return v + '0';
        else if (v == 256)
            return 0;
        else
            return v - 10 + 'A';
    }

    size_t v;
};
```

大数类设计与构建

- 更进一步，我们就可以在我们已分配的size_t数组的地址上原地构造NumToken完成ascii到真实值的转换，同时也可以将size_t转换为NumToken并调用toASCII()方法将真实值转换为ascii，构造代码如下

```
//通过string构造Num类，f为进制
Num makeNum(const std::string& s,char f)
{
    if (s.empty())
    {
        std::cerr << "Number is empty!" << std::endl;
        abort();
    }

    std::cout << s << std::endl;

    //符号识别
    bool has_sign = false;
    bool sign = true;
    if (s[0] == '-')
        has_sign = true, sign = false;
    else if (s[0] == '+')
        has_sign = true;

    Num result(s.size() - (has_sign ? 1 : 0), f, sign);
    const char* temp = s.c_str() + (has_sign ? 1 : 0);
```

```
    for (size_t i = 0; i < result.length; i++)
    {
        //在分配好的地址构造NumToken，实现将ascii转换为真实值
        new (result.v + i) NumToken(temp[i]);
        if (!(result[i] < f))
        {
            //防止\r\n换行导致将\r识别为非法
            if(temp[i]=='\r'&&i==result.length-1)
            {
                result.length=i;
                break;
            }

            for (size_t j = 0; j < i; j++)
                std::cerr << ' ';
            if(has_sign)
                std::cerr << ' ';
            std::cerr << '^';

            std::cerr << "\nNumber format error!\n";
            abort();
        }
    }

    return result;
}
```

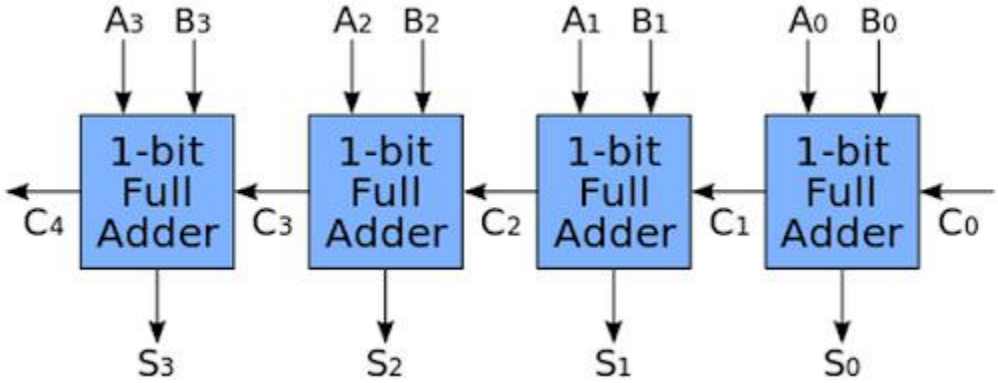
纲要

深蓝学院
shenlanxueyuan.com

- 第一部分：整体思路
- 第二部分：输入合规检查
- 第三部分：数值类构建
- **第四部分：大数加法**
- 第五部分：进制转换

大数加法

- 首先考虑加法器原理，我们可以参照加法器的原理自己实现一个抽象加法器，它可以接受任意相同进制的1位数相加减，并能够处理上一位的进位并产生进位，我们需要根据大数的位数连接足够多的加法器，而递归刚好满足要求



大数加法

- 递归开始前需要对符号做处理，代码如下

```
Num operator+(const Num& a,const Num& b)
{
    if (a.positive == b.positive)   //同号分发
    {
        Num result(std::max(a.length, b.length) + 1, a.format, a.positive);
        result[0] = _add<'+', '+'>(a.v, a.length, b.v, b.length, result.v + 1, result.format);
        return result;
    }
    else//异号分发
    {
        Num result(std::max(a.length, b.length), a.format, true);
        int temp = 0;
        if (a.positive)
            temp = _add<'+', '-'>(a.v, a.length, b.v, b.length, result.v, result.format);
        else
            temp = _add<'-', '+'>(a.v, a.length, b.v, b.length, result.v, result.format);
        //这里是一个特殊处理，如果进位为-1则这个数是一个补码，则需要进行转换
        if (temp == -1)
        {
            result.positive = false;
            for (size_t i = 0; i < result.length - 1; i++)
                result[i] = result.format - 1 - result[i];
            result[result.length - 1] = result.format - result[result.length - 1];
        }
        return result;
    }
}
```

大数加法

- 递归实现

```
//此函数实现广义加法，通过模板参数分发算子，内部通过递归实现整个加法，递归深度为最长的大数
template<char sign1,char sign2>  <T> 提供 IntelliSense 的示例模板参数
int _add(size_t* a, size_t a_length, size_t* b, size_t b_length, size_t* result, char format)
{
    int temp;
    //递归设计思想为 递归函数处理两操作数当前位和递归函数返回的进位结果之和并返回新的进位信息，将递归展开结构和加法器是一摸一样的，未结束的递归函数为全加器，满足结束条件时为半加器
    if (a_length == 1 && b_length == 1) //终止条件
        temp = __add<sign1, sign2>(*a, *b);
    else if (a_length == b_length)  //普通情况
        temp = _add<sign1, sign2>(a + 1, a_length - 1, b + 1, b_length - 1, result + 1, format) + __add<sign1, sign2>(*a, *b);
    else if (a_length > b_length)//如果两数长度不等则在短的数前补零，下同
        temp = _add<sign1, sign2>(a + 1, a_length - 1, b, b_length, result + 1, format) + __add<sign1, sign2>(*a, 0);
    else if (a_length < b_length)
        temp = _add<sign1, sign2>(a, a_length, b + 1, b_length - 1, result + 1, format) + __add<sign1, sign2>(0, *b);

    //处理正负进位
    if (temp < 0)
    {
        *result = temp + format;
        return -1;
    }
    else
    {
        *result = temp % format;
        return temp / format;
    }
}
```

```
//模板分发算子
template<char, char>
inline int __add(int a, int b);
template<>
inline int __add<'+', '+'>(int a, int b) { return a + b; }
template<>
inline int __add<'+', '-'>(int a, int b) { return a - b; }
template<>
inline int __add<'-', '+'>(int a, int b) { return b - a; }
```

纲要

深蓝学院
shenlanxueyuan.com

- 第一部分：整体思路
- 第二部分：输入合规检查
- 第三部分：数值类构建
- 第四部分：大数加法
- **第五部分：进制转换**

进制转换

- 进制转换过程即是对大数连续求余的过程，求余操作通过大数类中的%符号重载实现
- 右图演示了连续求余进行进制转换的代码实现

```
//格式转换，对原数不断求余得到
Num reFormat(Num& old, char new_f)
{
    size_t l = old.length * 6;
    size_t* new_v = new size_t[l];
    l = 0;
    //得到低位在前大数序列
    while (!old.isZero())
        new_v[l++] = old % new_f;

    Num result(l, new_f, old.positive);
    //反向储存
    while (l--)
        result[result.length - l - 1] = new_v[l];

    delete[] new_v;
    return result;
}
```

进制转换

深蓝学院
shenlanxueyuan.com

- 有图展示了求余的代码实现，通过大数类中的%符号重载实现，本质是模拟竖式除法过程
- 求余具体方法为从最高位对每一位求整数商，将商作为结果从高到低保留，将余数乘进制与下一位相加做为下一位的真实值，按此过程对每一位进行计算，最后一位的余数即为所求余数

```
//求余，进制转换输出核心，需要注意的是，求余也会将自身变为商，和正常的求余略有不同
unsigned char operator%(unsigned char f)
{
    //内部实现进制提升
    if (f > format) { ... }
    //找到第一个非零数
    size_t start_p = 0;
    for (size_t i = 0; i < length; i++)
        if (v[i] != 0)
        {
            start_p = i;
            break;
        }

    //对每一位执行除法，商保留为当前位结果，余数乘进制放入下一位除
    size_t new_length = length - start_p;
    size_t* new_v = new size_t[new_length];
    unsigned char result=0;
    for (size_t i = 0; i < new_length; i++)
    {
        size_t temp = result * format + v[start_p + i];
        new_v[i] = temp / f;
        result = temp % f;
    }

    delete[] v;
    v = new_v;
    length = new_length;
    return result;
}
```

深蓝学院
shenlanxueyuan.com

感谢各位聆听！

Thanks for Listening