深蓝学院
shenlanxueyuan.com

C++ 基础与深度解析
Project2思路解析提示

助教 天哲

整体思路

- 命令行参数/读取文件
- 长整数加减法
- 进制转换
- 其他

读取文件 （群里有）

深蓝学院 shenlanxueyuan.com

- 命令行参数：
- 针对int main(int argc, char *argv[]); argc表示命令行参数的个数。argv表示命令行参数内容。
- 打开文件：
- 使用ifstream进行文件操作，内容读取参考补充说明中的内容。
- 文件内容转换
  - 字符串转换：char (0-9) : int (48-57)、char (a-z) : int (97-122)、char (A-Z) : int (65-90)
  - 这里注意尽量不要使用magic number，使用 `int x= c-'a'` 这样可读性更好的表达方式
  - 字符检查：需要根据（基础部分需要检查是否小于10，扩展部分需要检查是否小于命令行参数输入的进制），同时删除最前面为0的数据。
- 注意：
  - 检查第一个符号是否是正负号
  - 检查是否是两行数据

计算处理

- 可以准备一些辅助函数，同时对读进的字符串进行一些预处理：
- 根据符号的异同判断，同号为加，异号为减
- 我自己可能会把符号“对齐”，就是会先把“0”表达为“+0”
- 去除头部的0（但是注意如果原来是“0”，不要也把它清除）
- 如果要原地处理，注意字符串的大小（例如 “999” + “1” 会需要4位）
- 注意：
- 小心处理加法的进位以及减法的借位
- 进行减法时，需要比较两个数组的尺寸大小

```
StatusCode BigNumAdd(std::string &num1, std::string &num2, const uint16_t base){
    StatusCode retVal = STATUS_OK;
    if (!num1.size() || !num2.size() || '-' == num1[0] || '-' == num2[0])
    {
        return STATUS_INTERNAL_ERROR;
    }
    // make sure length is enough
    if (num1.size() < num2.size())
        swap(num1, num2);
        // because a sum can't be longer than max length +1, we use num1 as return
    num1.insert(0, "0");
    uint16_t carry = 0;
    uint16_t oneSum = 0;
    size_t nonZeroIndex = std::string::npos;
    // start from tail to head
    auto it2 = num2.rbegin();
    for (auto it1 = num1.rbegin(); it1 != num1.rend(); it1++)
    {
        if (num2.rend() != it2)
        {
            oneSum = Char2Num(*it1) + Char2Num(*it2) + carry;
            carry = oneSum / base;
            *it1 = charMappers[oneSum % base];
            it2++;
        }
        else if (num2.rend() == it2) // it2 comes to an end
        {
            oneSum = Char2Num(*it1) + carry;
            carry = oneSum / base;
            *it1 = charMappers[oneSum % base];
            if (!carry) // if carry =0, we can take num from num1
                break;
        }
    }
    // cut leading zeros
    nonZeroIndex = num1.find_first_not_of("0");
    if (nonZeroIndex != std::string::npos)
        num1 = num1.substr(nonZeroIndex, num1.size() - nonZeroIndex);
    else
        num1 = "0";
    return retVal;
}
```

进制转换：长除法

模 n 取余法

4 进制数 $2103_{(4)}$ 转为 7 进制

	2 / 7 = 0	2 % 7 = 2
2 * 4 + 1 = 9	9 / 7 = 1	9 % 7 = 2
2 * 4 + 0 = 8	8 / 7 = 1	8 % 7 = 1
1 * 4 + 3 = 7	7 / 7 = 1	7 % 7 = 0
	1 / 7 = 0	1 % 7 = 1
1 * 4 + 1 = 5	5 / 7 = 0	5 % 7 = 5
5 * 4 + 1 = 21	21 / 7 = 3	21 % 7 = 0
	3 / 7 = 0	3 % 7 = 3

→ 111　　0

→ 3　　0

→ 0　　3

结束位标志

进制转换：长除法


```
StatusCode BaseConversionByDivid(std::string &str, std::string &res, const
uint16_t inputbase, const uint16_t outputbase)
{
    StatusCode retVal = STATUS_OK;
    if (str.empty() || inputbase > 36 || inputbase < 2 || outputbase > 36 ||
outputbase < 2)
    {
        retVal = STATUS_BAD_ARGUMENT;
        std::cerr << "BaseConversionByDivid has BAD ARGUMENTS\n";
        return retVal;
    }
    res.clear();
    std::string quotient = "";
    uint16_t remainder = 0;
    // till the end
    while (quotient != "0")
    {
        retVal |= BigNumDiv(str, outputbase, quotient, remainder, inputbase);
        res.push_back(charMappers[remainder]);
        str = quotient;
    }
    std::reverse(res.begin(), res.end());
    return retVal;
}
```

```
StatusCode BigNumDiv(std::string &dividend, const uint16_t divisor,
std::string &quotient, uint16_t &remainder, const int base)
{
    StatusCode retVal = STATUS_OK;
    if (dividend.empty() || divisor > 36 || divisor < 2 || base > 36 ||
base < 2)
    {
        retVal = STATUS_BAD_ARGUMENT;
        std::cerr << "BigNumDiv has BAD ARGUMENTS\n";
        return retVal;
    }
    quotient.clear();
    uint16_t sdividend = 0;
    uint16_t squotient = 0; // s stands for small
    for (auto &x : dividend)
    {
        sdividend = sdividend * base + Char2Num(x);
        squotient = sdividend / divisor;
        quotient.push_back(charMappers[squotient]);
        sdividend = sdividend % divisor;
    }
    remainder = sdividend;
    // cut leading zeros
    auto nonZeroIndex = quotient.find_first_not_of("0");
    if (nonZeroIndex != std::string::npos)
        quotient = quotient.substr(nonZeroIndex, quotient.size() -
nonZeroIndex);
    else
        quotient = "0";
    return retVal;
}
```

其他方法

可以先用vector<int> 来保存中间结果，这样中间计算比较方便，最后统一处理进位或者进制转换

例如，0xFF可以先表示为{15,15}
加法：
{15,15}+{1}={15,16}={16,0}={1,0,0}

进制转换（例如到10 进制）
{15}*16 （其实这里是循环做16次加法） = {2,4,0}
{2,4,0}+{15}={2,4,15}={2,5,5}

深蓝学院
shenlanxueyuan.com

感谢各位聆听！

Thanks for Listening