深蓝学院
shenlanxueyuan.com

项目三：编译期长整数加法

主讲人 陆一帆

长整数加法

- 把15章前4节的视频都看过去
- 编程实现长整数加法

1. 用结构体模板表示数组

```
template <unsigned int... args>
struct Cont {};
```

2. 实现翻转函数（好好看第4节的内容）

```
template <typename Res, typename Rem>
struct Flip;

template <unsigned int... Processed, unsigned int T, unsigned int... TRemain>
struct Flip<Cont<Processed...>, Cont<T, TRemain...>> {
    using type = typename Flip<Cont<T, Processed...>, Cont<TRemain...>>::type;
};

template <unsigned int... Processed, unsigned int T>
struct Flip<Cont<Processed...>, Cont<T>> {
    using type = Cont<T, Processed...>;
};
```

长整数加法

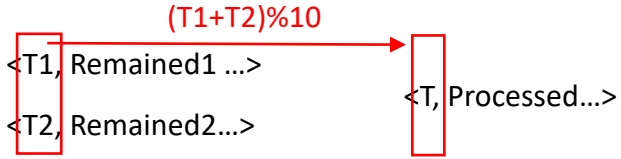
深蓝学院 shenlanxueyuan.com

●加法逻辑

1. 本质上还是个循环处理逻辑：不断把两个长整数的第N位上的数值（和上一个循环中产生的进位）相加对10取余得到结果中第N位的值，并记录是否产生进位（相加结果大于10），用于下一次循环。

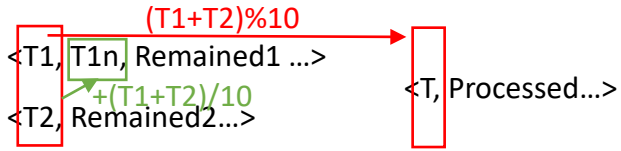
第一次尝试：

<Processed…, T, Remained…> --> <Processed…, T1, Remained1…, T2, Remained2…>



第二次尝试：

<Processed…, T1, Remained1…, T2, Remained2…> --> <Processed…, T1, T1n, Remained1…, T2,Remained2…>



长整数加法

●加法逻辑

第二次尝试扩展：

<Processed…, T1, T1n, Remained1…, T2, T2n, Remained2…>

<Processed…, T1, T1n, Remained1…, T2, Remained2…>

<Processed…, T1, T1n, Remained1…>

<Processed…, T1, Remained1…, T2, T2n, Remained2…>

<Processed…, T2, T2n, Remained2…>

<Processed …, T1, T2>

<Processed…, T1>

<Processed…, T2>

针对每种形式，区分产生进位和没产生进位两种结果，用requires方法（第3节内容）实现：

```
template <unsigned int... Processed, unsigned int T1, unsigned int T2>
    requires (T1 + T2 >= 10)
struct Add_<Cont<Processed...>, Cont<T1>, Cont<T2>> {
    using type = Cont<Processed..., (T1 + T2) % 10, 1>;
};
```

```
template <unsigned int... Processed, unsigned int T1, unsigned int T2>
    requires (T1 + T2 < 10)
struct Add_<Cont<Processed...>, Cont<T1>, Cont<T2>> {
    using type = Cont<Processed..., (T1 + T2) % 10>;
};
```

长整数加法

深蓝学院
shenlanxueyuan.com

●顺序执行整个过程

1. 参照第2节的内容顺序执行所有逻辑即可

```
template <typename Res, typename Rem1, typename Rem2>
struct AddFun {
    using filped1 = typename Flip<Cont<>, Rem1>::type;
    using filped2 = typename Flip<Cont<>, Rem2>::type;
    using res = typename Add_<Cont<>, filped1, filped2>::type;
    using res2 = typename Flip<Cont<>, res>::type;
    using value = typename RmZero<Cont<>, res2>::type;
};

// 别名模版
template<typename Res, typename Rem1, typename Rem2>
using Add = typename AddFun<Res, Rem1, Rem2>::value;
```

2. Print函数助教有讲过，不再赘述了

3. 调用

```
using input1 = Cont<9, 9>;
using input2 = Cont<0, 0, 9, 9>;
using res = Add<Cont<>, input1, input2>;
```

进制转换

●与Proj1相同的部分

Print函数实现；Flip函数实现；数组的表示

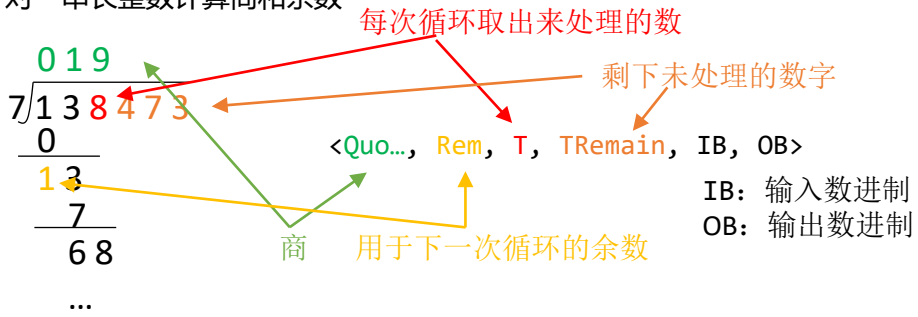
●复习第一次作业中的进制转换算法

1. 非长整数情况：将N进制数值转换为10进制，用转换后的10进制数值不断除以M，将得到的余数记录在数组中，将商继续除以M；循环下去，直到商为0；将余数数组翻转后，所得值即为M进制的结果。

2. 长整数情况：由于长整数过大，无法一次性转换为一个10进制的int型的变量执行上面的操作，所以得通过循环的方式计算商和余数。

●代码实现

1. 对一串长整数计算商和余数



进制转换

●代码实现

1. 对一串长整数计算商和余数（循环逻辑）

```
template <unsigned int... Quo, unsigned int Rem, unsigned int T, unsigned int... TRemain, unsigned int IB, unsigned int OB>
struct CalRemAndQuo<Cont<Quo...>, Cont<Rem>, Cont<T, TRemain...>, Cont<IB>, Cont<OB>> {
    using rem = typename CalRemAndQuo<Cont<Quo..., (Rem * IB + T) / OB>, Cont<(Rem * IB + T) % OB>, Cont<TRemain...>, Cont<IB>, Cont<OB>>::rem;
    using quo = typename CalRemAndQuo<Cont<Quo..., (Rem * IB + T) / OB>, Cont<(Rem * IB + T) % OB>, Cont<TRemain...>, Cont<IB>, Cont<OB>>::quo;
};
```

2. 对一串长整数计算商和余数（循环结束逻辑）

3. 注意循环结束的时候有个RmZero函数需要实现，用于将商最前面的“0”都删除掉，这和Flip操作类似，不再赘述了。

进制转换

深蓝学院 shenlanxueyuan.com

●代码实现

1. 对每次得到的商再执行前面的长整数除法运算，直到商为0(商所对应的数组为空)；将每次得到的余数保存在数组里；

```
template <unsigned int... Rems, unsigned int Rem, unsigned int T, unsigned int... Input, unsigned int IB, unsigned int OB>
struct CalRem<Cont<Rems...>, Cont<Rem>, Cont<T, Input...>, Cont<IB>, Cont<OB>> {
    using rem = typename CalRemAndQuo<Cont<>, Cont<0>, Cont<T, Input...>, Cont<IB>, Cont<OB>>::rem;
    using quo = typename CalRemAndQuo<Cont<>, Cont<0>, Cont<T, Input...>, Cont<IB>, Cont<OB>>::quo;
    using rems = typename CalRem<Cont<Rems..., Rem>, rem, quo, Cont<IB>, Cont<OB>>::rems;
};
```

2. 别忘了加上循环退出逻辑

3. 最后把rems翻转一下，就得到M进制的输出结果了，翻转函数前面有讲，不赘述啦

在线问答

深蓝学院
shenlanxueyuan.com

Q&A

深蓝学院
shenlanxueyuan.com

感谢各位聆听！

Thanks for Listening