# 说明

## 运行环境

### 编译器(g++)版本

### 

* 1. 编译命令

g++ % -g -o %< -Wall -std=c++11

## 代码说明

代码分为四部分（由于最后只能交一个cpp文件），分别为

### 链表部分（这样写的好处：可适用于任何数据类型的节点，比如下面用的“多项式的一项”这种数据类型）

从代码标注“LinkList.h”注释处开始，包含

#### Node节点

* + - 1. data数据（自定义）
      2. next 下一个指针
      3. 三个构造函数
    1. LinkList 链表
       1. head头指针
       2. 构造函数
       3. insert\_in\_front(T \_data) 插入在前
       4. insert\_in\_end(T \_data) 插入在后
       5. reverse() 就地反转

### 多项式部分

从代码标注“Polynomial.h”注释处开始，包含

#### PolynomialTerm多项式的一项

* + - 1. exp 指数（只能为int）
      2. coef 系数（自定义，类型记做T，这样写的好处：无论是int还是double还是别的数据类型，都能适用）
      3. 构造函数
      4. 重载输出流（这样写的好处：可以通过cout直接输出）
    1. Polynomial 多项式：继承自链表
       1. quick\_pow(T base, int exp) 快速幂函数（这样写的好处：在O(log exp)内算出base^exp）
       2. 构造函数
       3. calc(T x) 带入x计算（x的类型同coef的类型）
       4. 重载+（这样写的好处：可以通过a+b直接计算）
       5. 重载输出流（这样写的好处：可以通过cout直接输出）
       6. 重载输入流（这样写的好处：可以通过cin直接输入）
  1. 菜单部分

从代码标注“Menu.h”注释处开始，包含

#### Menu 菜单

* + - 1. opt 操作编号
      2. map< string, Polynomial<T> > mp（多项式名-多项式 映射，这样写的好处：方便储存与读取多项式）
      3. display() 输出菜单
      4. main\_loop() 主循环（不断读取opt，执行相关操作）

其中:

Opt 1. 新建多项式（输入多项式名和多项式）

Opt 2. 输出多项式（输入多项式名，输出多项式）

Opt 3. 反转多项式（输入多项式名，就地反转多项式）

Opt 4. 带入x计算（输入多项式名和x，计算多项式的值）

Opt 5. 多项式相加（输入两个多项式名，计算这两个多项式的值）

* 1. 主函数部分

从代码标注“main.cpp”注释处开始，包含

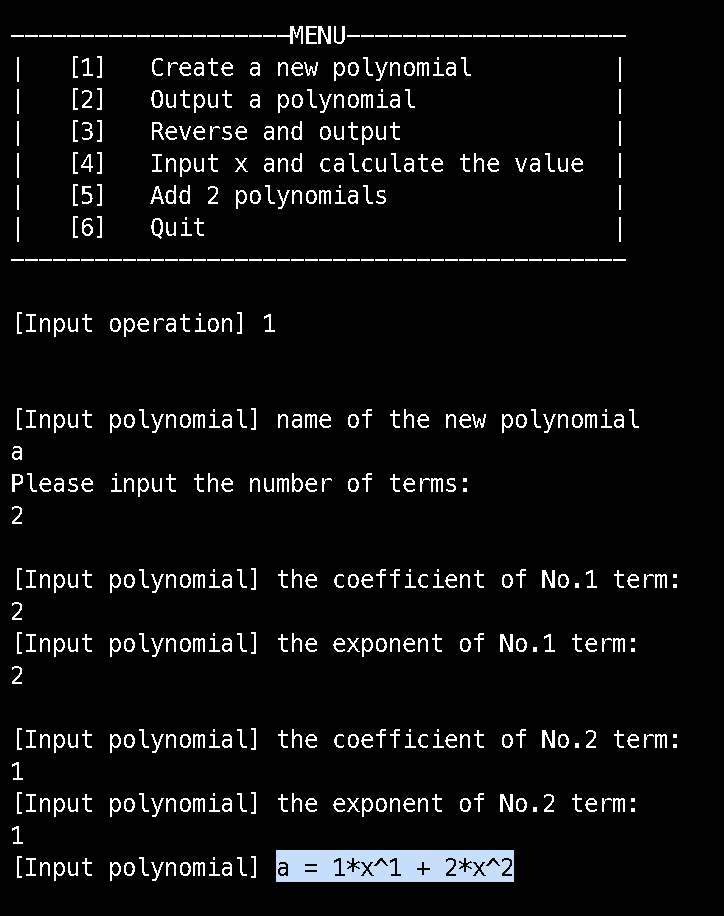
#### main() 主函数

* + - 1. Menu<int> menu 新建菜单，指定系数类型为int（如果测试别的数据类型直接改变尖括号里的数据类型）
      2. menu.main\_loop() 开始循环

## 测试情况

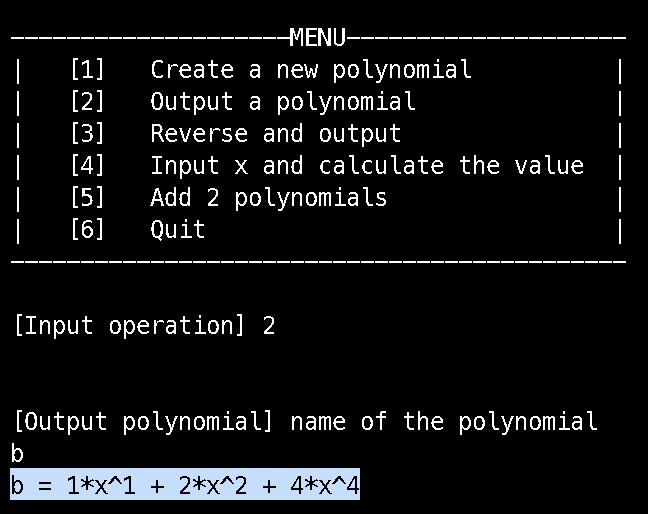
### 从键盘读入一元多项式中每一项的系数和指数，建立带表头结点的单链表存放一元多项式(按照指数升序排列)

菜单选择1，输出多项式名，输入多项式项数，输入每一项的系数和次数（可乱序输入）。此处输入a = 1\*x^1 + 2\*x^2和 b = 1\*x^1 + 2\*x^2 + 4\*x^4，结果如下：



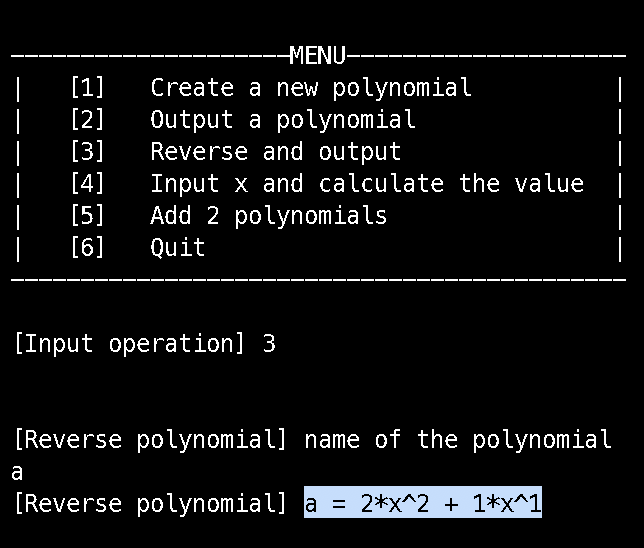
### 输出一元多项式的所有数据元素(按照指数升序输出每一系数非0项的系数和指数);

菜单选择2，输入多项式名。此处输出b，结果如下：



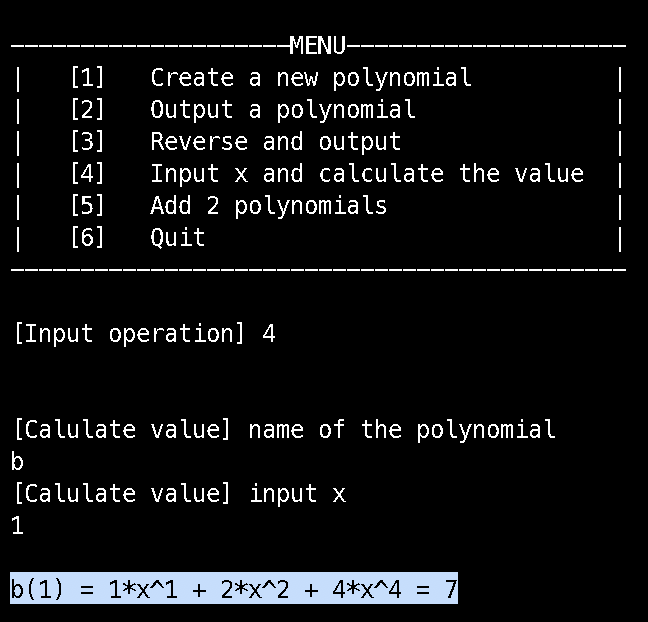
### 将单链表存放的一元多项式就地逆置，变成按照指数降序排列

菜单选择3，输入多项式名。此处输入a，结果如下：



### 输入自变量的值，计算一元多项式的值(设计高效算法)

菜单选择4，输入多项式名和x的值。此处输入b，x=1，结果如下：



### 求2个一元多项式的和多项式

菜单选择5，输入两个多项式。此处输入a和b，结果如下：

