

四则运算 core 第 7 组 API 文档

简要描述：

- 四则运算题目随机生成器

类名：

- arithmetic

类的成员变量：

变量名	类型	默认值	说明
expNum	int	1	生成表达式的数目
expType	int	0	表达式类型，0 代表整数，1 代表小数，2 代表分数，或用宏定义 INTEGER，DECIMAL，FRACTION
oprNum	int	1	运算符数目
oprType	int	0	运算符类型，0 代表 "+-"，1 代表 "+-*/"，2 代表 "+-*/^"，或用宏定义 ADDSUB，MULDIV，POW
min	int	1	操作数的最小值（表达式类型为分数时为分母最小值）
max	int	10	操作数的最大值（表达式类型为分数时为分母最大值）
accuracy	int	2	表达式类型为小数时，显示结果的精度

类的成员函数：

函数声明	说明
void setExpNum(int n)	设置生成表达式数目
void setExpType(int n);	设置生成表达式类型
void setOprNum(int n);	设置运算符数目

函数声明	说明
void setOprType(int n);	设置运算符种类
void setBounds(int min, int max);	设置操作数范围
int getExpNum();	返回表达式数目
void generate();	生成表达式
string* getExpSet();	返回生成的表达式 (字符串数组)
string* getAnsSet();	返回与表达式对应的结果 (字符串数组)

调用示例：

```

{
    arithmetic test;                //定义一个 arithmetic 类变量

    test.setExpNum(20);              //生成 20 个表达式
    test.setExpType(DECIMAL);        //表达式类型为小数
    test.setBounds(1, 20);           //操作数范围为 1~20
    test.setOprNum(4);               //运算符为 4 个
    test.setOprType(MULDIV);         //运算符包括 "+-*/"
    test.setAccuracy(2);             //结果显示两位小数

    test.generate();                 //生成表达式

    string* expSet;                  //表达式数组
    string* ansSet;                  //结果数组
    expSet = test.getExpSet();
    ansSet = test.getAnsSet();
    for (int i = 0; i < test.getExpNum(); i++) {
        //屏幕输出表达式及结果
        cout << expSet[i] << " = " << ansSet[i] << endl;
    }
}
}

```

DLL 使用步骤：

- 工程中存在 cpp 的情况下，修改项目属性：属性--C/C++--预处理器--预处理器命令--添加_CRT_SECURE_NO_WARNINGS (由于使用了 sprintf 函数)
- 将 7_ArithmeticDll.dll , 7_ArithmeticDll.lib 以及 7_ArithmeticDll.h 三个文件 复制到存放即将调用 core 的.cpp 的文件夹中
- 在头文件中添加 现有项 "7_ArithmeticDll.h"
- 在资源文件中 添加 现有项 "7_ArithmeticDll.lib"
- 在用来调用 core 的.cpp 中添加 #include"7_ArithmeticDll.h"
- release/debug、x64/x86 不能混用