20220602-元编程

1.学习内容

1.1 元编程

编译期常量从何而来

- sizeof操作符

C++ 复制代码

```cpp
class someClass
{
};
int const count=10;//作为数组的size，编译期常量
someClass theMovie[count]={};//常量表达式，在编译期计算
int const othercount=26;//只是常量，非编译期常量

int i=430;
unsigned char buffer[sizeof[i]]={};//常量表达式，在编译期计算
```

- 静态类成员变量

```cpp
struct someStruct
{
    static unsigned const size1=44;//编译期常量
    enum {size2=45};//编译期常量
    int someInt[size1];//常量表达式，在编译期运行
    double someDouble[size2];//常量表达式，在编译期运行
}
```

编译期常量表达式：任何不是用户自己定义，必须通过编译期计算出来的字面量都属于编译期常量表达式。并非所有的常量表达式都是编译期常量表达式，只有要求编译器计算时才是。

```cpp
const int i=100;
const int j=i*200;//常量表达式

const int k=100;
const int p=k*200;//编译期常量表达式，由下边的数组确定
unsigned char helper[p]={};
```

编译期运算

有时可以通过某些手段胁迫编译器，把运算任务从运行时提前到编译期。

```cpp
const int doubleCount=10;
unsigned char doubleBuffer[doubleCount*sizeof(double)]={};
```

sizeof(void)==4可以用来判断一个系统是不是一个32位系统

C++ 复制代码

```cpp
std::string nonsense(char input)
{
    auto const index=(sizeof(void*)==4)?0:1;
    auto const someLabel="some"[index];
    switch(input){
    case someLabel:
        return "AA";
    default:
        return "BB";
    }
}
```

使用模板进行编译期运算

实例化模板的参数必须为编译期常数——编译器会在编译期计算作为实例化模板参数的常量表达式

C++ 复制代码

```cpp
template <unsigned N>
struct Fibonaci;

template<>
struct Fibonaci<0>
{
    static unsigned const value=0;
};

template<>
struct Fibonaci<1>
{
    static unsigned const value=1;
};

template<unsiged N>
struct Fibonaci
{
    static unsigned const value=Fibonaci<N-1>::value+Fibonaci<N-2>::value;
};
```

最后一个模板递归式地去实例化参数位N的模板，递归终止在模板参数为0和1时。

2.结果描述

今天依旧围绕元编程进行学习，学习内容不够多，时间没有利用好。明天继续。