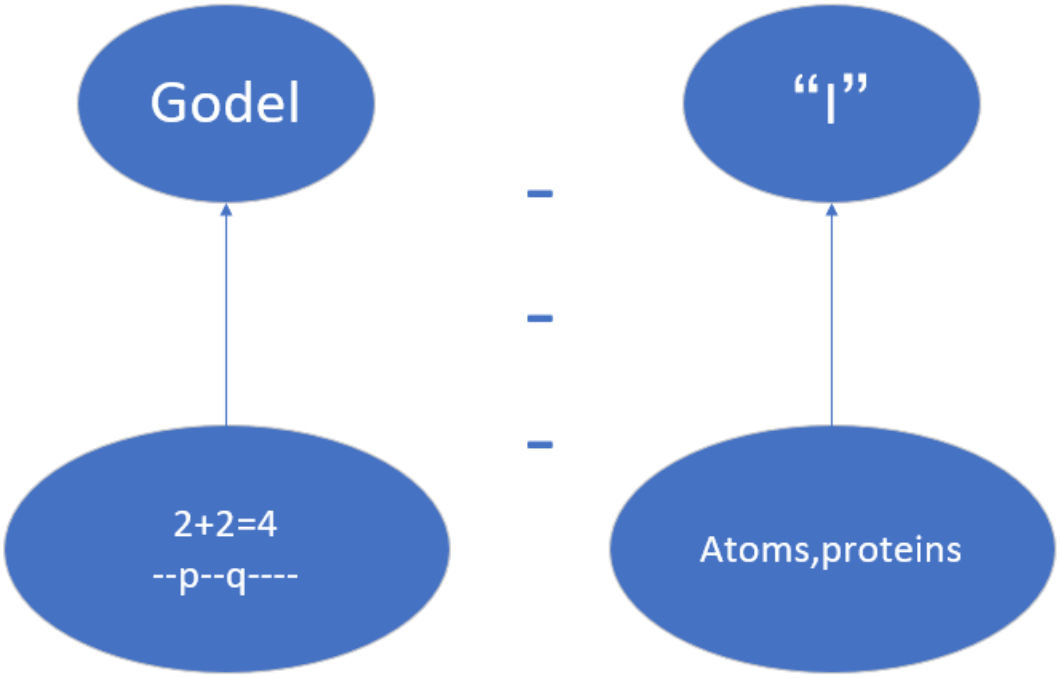
# 20220410-GEB&C++



# 1.过程描述

## 1.1 GEB

### Course 1：




#### 1） five important tools for thinking

- **Isomorphisms**
  - applies when 2 complex structures can be mapped onto each other in a ways that to each part of one structure there is a cooresponding part on the other sturcture where corresponding means they play similar roles and the respective structures
- **Recursion**
- **Paradox**
  - Verdical
  - Falsidical
  - Antinomy
    - This sentence is not true
    - Barber's paradox
- **Infinity**
- **Formal systems**
  - String: ordered sequence
  - Axiom: starting point
  - Theorem: String which results at the end of a derivation
  - Rules: rules of inference

### 2) Some important thinkings

- Hofstadter said:

> of course there are cases when only a rare individual will have a vision to  percive a system which governs many people's lives, a system which had never before even been recognized as a system, then such people often devote their lives to convincing other people that the system really is there and that it ought to be exited from

- Try to do meta thinking in a higher level and see is it worth being exiting that system

### 3) PQ system

- xp−qx−  is an axiom whenever  x={−−−,−−−−−,...}
  - −−p−q−−−
- xpyqz−>xpy−qz−

## 1.2 C++

模板 C++ 复制代码

```cpp
template<typename T>
void Print(T value)
{
    cout << value << endl;
}//the function only created when called

template<typename T,int N>
class Array
{
private:
    T m_Array[N];
public:
    int GetSize() const { return N; }
};

int main()
{
    Print<int>(50);
    Print(5.5f);

    Array<int,5> arraydemo;
    cout << arraydemo.GetSize() << endl;
}
```

stack和heap C++ 复制代码

```cpp
stack跟heap都在RAM中
allocating memory on the heap is a whole thing, whereas allocating memeory on stack is like one CPU instruction
try to allocate on the stack whenever possible

int main()
{
    //stack
    int value = 5;
    //heap
    int* hvalue = new int;
    *hvalue = 5;
    delete hvalue;
}
```

auto的用法 C++ 复制代码

```cpp
int main()
{
    vector<string> strings;
    strings.push_back("orange");
    strings.push_back("Apple");
    for (vector<string>::iterator it = strings.begin(); it != strings.end(); it++)
    {
        cout << *it << endl;
    }
    for (auto it=strings.begin(); it != strings.end(); it++)
    {
        cout << *it << endl;
    }

}
-------------------------------------------------
class Device
{
};
class DeviceManager
{
private:
    unordered_map<string, vector<Device*>> m_Devices;
public:
    const unordered_map<string, vector<Device*>>& GetDevices() const
    {
        return m_Devices;
    }
};

int main()
{
    DeviceManager dm;
    //the massive way
    const unordered_map<string, vector<Device*>>& devices1 = dm.GetDevices();
    //using way
    using DeviceMap = unordered_map<string, vector<Device*>>;
    const DeviceMap& devices2 = dm.GetDevices();
    //typedef way
    typedef unordered_map<string, vector<Device*>> DeviceMapMap;
    const DeviceMapMap& devices3 = dm.GetDevices();
    //auto way
    const auto& devices4 = dm.GetDevices();
```

```
}
```

函数指针 C++ 复制代码

```cpp
void Hello(int a)
{
    cout << "Hello" <<a << endl;
}

void PrintValue(int value)
{
    cout << "Values: " << value << endl;
}

void ForEach(const vector<int>& values,void(*func)(int))
{
    for (int value : values)
        func(value);
}
int main()
{
    //way1
    void(*cherno)(int) = Hello;

    //way2
    auto function = Hello;//函数名是指针

    //way3
    typedef void(*Hellofunction)(int);
    Hellofunction func = Hello;
    func(8);


    vector<int> values = { 1,5,4,3,6 };
    ForEach(values, PrintValue);

    ForEach(values, [](int value) {cout << "Value: " << value << endl;
});
}
```

lambda C++ 复制代码

```cpp
void PrintValue(int value)
{
    cout << "Value: " << value << endl;
}

void ForEach(const vector<int>& values,const function<void(int)>& func)
{
    for (int value : values)
        func(value);
}
int main()
{
    vector<int> values = { 1,5,4,3,6 };
    //lambda在findif中的应用
    auto it=find_if(values.begin(), values.end(), [](int value) {return value > 3; });
    cout << *it << endl;

    //函数指针
    ForEach(values, PrintValue);

    //lambda的用法
    int a = 6;
    auto lambda = [=](int value)//[]pass 外面的variable的方式，有=，&，a(特定的变量)
    {
        cout << "Value: " << value << endl;
    };
    ForEach(values,lambda);

}
```

# 2.结果输出

今天在MIT open course上找到了很多想学的课（焦虑感的外显），主要是CS相关的。第一门课选的是GEB，这其实是一本书，据说集科学之大成，MIT为此开了一门课我是没想到的。去年差不多这个时候有稍微看过一点，但因为比较艰深，所以当时也没能坚持下去。现在正好边听课边学习一下。下午主要看了The cherno的C++视频，相较于前期比较基础的内容，后面的50个视频主要cover一些比较分散、相对高级的C++语法，所以学起来感觉效率不高（还是存在心不在焉的情况）。争取明后两天把视频看完。