**作业02 游戏外挂的内存搜索核心代码**

**一、背景**

当我们编写游戏外挂时，需要在游戏内存中搜索特定数值的内存地址（指针）。我们一般只能从游戏显示中看到这一特定数值（比如，生命值显示为99），但不能直观分辨该数值在内存中是以何种类型存储的（int的99？float的99？double的99？）。故此，需要在游戏内存的一定范围内，同时搜索不同该数值的不同编码形式（不同码长和不同编码机制）对应的内存地址。

**二、重要前提**

某种数据类型的变量（包括数组元素，类的数据成员），在内存中的存储位置一定是该类型字节数的整数倍。如，所有int和float类型的变量，在内存里的存储地址（指针值）均为4的整数倍；double类型的变量，在内存里的存储地址（指针值）均为8的整数倍。

**三、要求**

完成以下代码，实现内存搜索功能。

bool SearchInMemory(

int Value,

void\* MemoryBegin,

size\_t SizeInByte,

vector<int\*>& IntPtrList,

vector<float\*>& FltPtrList,

vector<double\*>& DblPtrList

);

* Value为从游戏显示中读到的数值；
* MemoryBegin为指向游戏内存需搜索区域的指针，其值一定是8的整数倍；
* SizeInByte为游戏内存需搜索区域的字节数，size\_t是C/C++中表示字节数的类型，在32位编译中size\_t是unsigned类型，用于指明数组长度或下标。
* 其余三个vector均为搜索结果，调用函数时均为空，分别存储所有：内存中值可显示为Value的存储地址（指针），int型Value的指针存储于IntPtrList，float型Value的指针存储于FltPtrList，double型Value的指针存储于DblPtrList；因为游戏内存中可能存在多个可显示为Value的内存区块，故要用Vector存储，以便后续筛选。

1. 返回值，当搜索结果为空时返回false，否则为true
2. 请注意，不要遗漏待搜索区域，同时注意搜索效率
3. 文件命名：学号\_02\_01\_SearchInMemory.cpp和学号\_03\_01\_ SearchInMemory.hpp
4. 需要足够帮助阅读者理解的注释，包括数据类型释义和执行代码解释
5. 需引用vector头文件