20220411-GEB&C++

1.过程描述

1.1 GEB

Course 2：

1）Recursion

Course 3：

1）Three terms

2）Godel's Theorem

3）Some interesting points

1.2 C++

2.结果输出

1.过程描述

1.1 GEB

Course 2：

1）Recursion

- **Recursive function is a function that calls itself**
- **Example:**
  - Koch Curve
  - Sierpinski triangle
  - Barnsley fern
  - Mandelbrot set

Course 3：

1）Three terms

- **Consistency**
- **Completeness**
- **Geometry**

2）Godel's Theorem

- any system as powerful as Number Theory which can prove its own consistency: that system is necessarily inconsistent
- any system as powerful as Number Theory is necessarily incomplete

3）Some interesting points

- Genius is 1，1+1/2+1/4+1/8+...=2，another genius born
- The interesting thing is we can catch infinite thing in a very finite way, like you can map infinite numbers out from between 0 to 1.
- IBM's deep blue is an example of recursion.

1.2 C++

thread C++ 复制代码

```cpp
#include <thread>
static bool s_Finished = false;
void DoWork()
{
    using namespace std::literals::chrono_literals;
    std::cout << "Started thread id=" << std::this_thread::get_id() << std::endl;
    while (!s_Finished)
    {
        std::cout << "Working" << std::endl;
        std::this_thread::sleep_for(1s);
    }
}

int main()
{
    std::thread worker(DoWork);
    std::cin.get();
    s_Finished = true;
    worker.join();
    std::cout << "Finished" << std::endl;
    std::cout << "Started thread id=" << std::this_thread::get_id() << std::endl;
    std::cin.get();
}
```

Timing C++ 复制代码

```cpp
#include <chrono>
#include <thread>

struct Timer
{
    std::chrono::time_point<std::chrono::steady_clock>start, end;
    std::chrono::duration<float> duration;
    Timer()
    {
        start = std::chrono::high_resolution_clock::now();
    }
    ~Timer()
    {
        end = std::chrono::high_resolution_clock::now();
        duration = end - start;
        float ms = duration.count() * 1000.0f;
        std::cout << "Timer took " << ms << "ms" << std::endl;
    }
};

void func1()
{
    Timer timer;
    for (int i = 0; i < 100; i++)
    {
        std::cout << "Hello" << std::endl;
    }
}

void func2()
{
    Timer timer;
    for (int i = 0; i < 100; i++)
    {
        std::cout << "Hello\n";
    }
}

int main()
{
    func1();
    func2();
    std::cin.get();
}
```

Multidimensional Arrays C++ 复制代码

```cpp
int main()
{
    int** a2d = new int* [5];

    for (int i = 0; i < 5; i++)
    {
        a2d[i] = new int[5];
    }

    for (int y = 0; y < 5; y++)
    {
        for (int x = 0; x < 5; x++)
        {
            a2d[x][y] = 5;
        }
    }

    for (int i = 0; i < 5; i++)
    {
        delete[] a2d[i];
    }
    delete[] a2d;
    /*
    int* array = new int[5 * 5];
    for (int y = 0; y < 5; y++)
    {
        for (int x = 0; x < 5; x++)
        {
            array[x + y * 5] = 2;
        }
    }
    */
    int*** a3d = new int** [50];
    for (int i = 0; i < 50; i++)
    {
        a3d[i] = new int* [50];
        for (int j = 0; j < 50; j++)
        {
            int** ptr = a3d[i];
            ptr[j] = new int[50];
        }
    }
    a3d[0][0][0] = 0;

    std::cin.get();
```

```
}
```

sorting C++ 复制代码

```cpp
#include <algorithm>
#include <functional>
int main()
{
    std::vector<int> values = { 3,5,7,2,1,8};
    std::sort(values.begin(), values.end(),std::greater<int>());//降序
    std::sort(values.begin(), values.end(), [](int a, int b)
        {
            return a < b;//升序
        });
    std::sort(values.begin(), values.end(), [](int a, int b)
        {
            if (a == 1)
                return false;
            if (b == 1)
                return true;
            return a < b;//升序，1放在最后面
        });

    for (int value : values)
    {
        std::cout << value << std::endl;
    }
    std::cin.get();
}
```

type punning C++ 复制代码

```cpp
struct Entity
{
    int x, y;
    int* Getposition()
    {
        return &x;
    }
};
int main()
{
    Entity e = { 5,8 };
    int* position = (int*)&e;
    int* pos = e.Getposition();
    std::cout << position[0] << "," << position[1] << std::endl;
    std::cout << pos[0] << "," << pos[1] << std::endl;
    std::cin.get();
}
```

virtual destructor C++ 复制代码

```cpp
class Base
{
public:
    Base() { cout << "Base constructor\n"; }
    virtual ~Base() { cout << "Base destructor\n"; }
};

class Derived:public Base
{
public:
    Derived() { m_Array = new int[5]; cout << "Derived constructor\n"; }
    ~Derived() { delete[] m_Array; cout << "Derived destructor\n"; }
private:
    int* m_Array;
};
int main()
{
    Base* base = new Base();
    delete base;
    cout << "------------" << endl;
    Derived* derived = new Derived();
    delete derived;
    cout << "------------" << endl;
    Base* demo = new Derived();
    delete demo;//, 如果没有加virtual, 则没有call derived deconstructor;会导致memory leak
}
```

2.结果输出

今天看了两个GEB的lecture，大致的内容能get到，但一些具体的细节没有深入思考。下午看了普林斯顿大学一门关于中国政治体制的课，感觉还挺有意思的。晚上主要看了The cherno的C++视频，只看了不足十个，明天要完结估计很困难。Anyway，努力努力。