# 14级数据结构C/C++补充内容大纲

姚尧
Tel: 185-2023-9106
Email: whuyao@foxmail.com

- 作业中存在的问题
  - char*字符串结尾 '\0'
  - 尽量不要使用隐式转换
  - 代码中尽量不要出现中文
  - 浮点型不可能相等
    - if (abs(a-b) < 1e-6) return true;
  - 如何四舍五入
    - int nX = int(dX + 0.5);
  - 类的默认构造函数和析构函数尽量不要省略
  - 指针的大量问题
  - 切莫浮沙筑高台
- 内存和数据结构
- 引用
  - 基本引用
    - int b;
    - int &a = b;
  - 函数引用
    - 引用用作函数的返回值
      - bool MyAlogorithmFunction(type input_params, type& output_results);
    - 用const 保护实参不被修改 int fun(const int& _nParam);
- 指针基本概念
  - 内存：线性存储结构
  - 指针基本概念
    - 指针的值
    - 指针的类型
      - 指针类型的强制改变会导致指向的空间也发生改变
      - void*作为函数形参传递

        - unsigned char* => 其他类型 （ 请参考 c指针重要的补充内容 ）
      - 指针和所指向的位置关系
    - 指针的定义： int* p1, *p2;
    - 指针的运算
      - 赋予的应该是内存中存在的一个地址
      - 和变量类型完全一致
      - 置NULL和销毁指针的习惯
      - 指针+1或-1
    - 什么情况下算释放了内存而且没有野指针？*(*普通指针*)*
- 数组和指针
    - 数组名就是一个常量指针，同时指向数组头
    - 指向数组元素的指针 p[i], *(p+i)
    - new和delete, (malloc/不初始化, calloc/初始化, realloc/某个位置开辟) & free 不能混用
    - delete时的注意事项，当指针位置改变的时候，不能执行delete
    - 指针的赋值方法
      - 直接赋值
      - memcpy
    - 二维指针的空间结构 创建和销毁方法
    - 什么情况下算释放了内存而且没有野指针？（ 数组和链表 ）
- 遥感数据中**BSQ**和**BIP**格式的的指针引用方法
    - BSQ (GDAL => geotiff, envi hdr, erdas img, et al.)
    - BIP (openCV => jpeg, bmp)
    - BIL (不常见)
- 指针用作函数的参数
    - 如果需要改变指针的指向数据的值，请采用“指针引用”
      - bool ChangeInputArrayValue(float*& pfData, int nDataCount);
    - 函数指针，回调函数(返回的是一个地址)：函数名本身就是函数的地址
      - 类型说明符 * 函数名(参数)
        - e.g. 1. int* getData(int a, int b) {static int c = a+b; return &c;}
        - e.g. 1. printf(“%d\n”, *getData(param1, param2));
        - e.g. 2. double print_area(double(*p)(double &x,double &y), double &x,double &y) {cout<<p(x,y)<<endl; return 0;}
        - e.g. 2. double mySum(double &x, double &y) {return x+y;}

        - e.g. 2. print_area(mySum, x, y);
- 类（应用方面补充内容）
    - 构造函数和析构函数
    - 在类中怎么使用指针  相关注意事项！(类指针的创建、使用和销毁)
    - 静态成员，外部成员等 (略)
    - 使用模板函数来获取数据
    - 尽量不要用友元和多重继承
- **C++标准库 (重点是看c++ primer)**
    - 参考C++ primer
    - IO类
    - 顺序容器：vector, list, deque, array, string
        - 迭代器 iterator
    - 泛型算法 generic algorithm
        - #include <algorithm>
        - #include <numeric>
        - sort(v.begin(), v.end(), *fun());
        - replace
        - accumulate(v.begin(). v.end(), init_value = 0); 求和
        - for_each
        - lambda函数
    - 关联容器：
        - 按照关键字存储：map set(关键字就是value) multimap multiset
        - 无序集合：unordered_map ... (用哈希函数组织存储的map)
    - 智能指针
        - shared_ptr 允许共享
        - unique_ptr 独占指针
- 栅格和矢量数据处理基础**C++库GDAL (提供材料请自学)**
    - 编译和配置GDAL: http://www.gdal.org/
    - 使用GDAL处理栅格处理
    - 使用OGR处理矢量数据