**《介绍》**

c++学习笔记

**《环境》**

**vscode**

使用vscode的优势在于vscode体量小、c++和python代码可以一起开发

vscode下载：https://code.visualstudio.com/

环境配置：下载mingw，在vscode插件中下载c++插件，搜教程配置编译环境。配置时注意使用g++替代gcc

MinGW：c++开发工具，包含GCC编译器，安装c++环境需要下载MinGW

MinGW下载地址：https://github.com/niXman/mingw-builds-binaries/releases

c\_cpp\_properties.json：配置好c++插件后会自动在.vscode下生成的环境配置文件

tasks.json：编译时的配置文件。自动生成：[终端菜单]->[配置文件]->[选择MinGW环境]。文件中的args参数的-g为debug编译(默认)，改为-02为release编译

编译：[终端菜单]->[运行生成任务]->[选择MinGW环境]。会生成.exe文件

运行：在下方终端中输入[.\文件名.exe]运行

vscode运行：点击[运行和调试]后会自动编译和运行，并且如果有断点会直接调试。快捷键[ctrl+f5]

新项目：创建新项目时，可以复制原来的.vscode配置文件

多文件编译：多文件编译(不同文件间函数调用)时要改造tasks.json

格式化：c++代码按ctrl+s保存时会自动格式化

**其他**

cmake：开源、跨平台的项目构建系统，支持多种操作系统和编译器。比如从github下载和安装库，可以通过cmake的方式构建来避免环境影响

**《c++基础》**

c++标准：c++语言规范，国际标准化组织制定，c++标准和编译器都在不断更新

c++库安装：由于c++的编译器没有统一的标准，导致很多库不通用，没有像python的conda一样的库管理工具，通常要手动下载和管理c++的库

编译器：c++语言需要编译器转换为对应系统的可执行文件，比如windows上编译后为.exe文件。常见的编译器有GCC、Clang、MSVS等。常用的编译器为GCC，支持多种系统和最新c++标准。通常会下载MinGW开发工具(包含GCC编译器)来配置c++环境

GCC：GCC编译器有gcc、g++两种，gcc针对c，g++针对c++

编译模式：编译器会将代码编译为可执行语言，有debug编译和release编译两种方式。前者包含调试信息，后者是经过大量优化的最终版本

字符串：c++中要使用双引号

静态库：.a或.lib。程序使用静态库时，会复制静态库的内容到系统上。优点是效率高，跨系统时不依赖环境；缺点是占用空间大，每次调试代码时要重新编译静态库

动态库：.so或.dll。动态库存在于程序外部，供程序调用

**名字空间**

一个名字空间包含多个类

namespace space//{...}：定义名字空间A。名字空间相当于一个函数库

using namespace space;：使用space中所有定义的常量或变量

space::A：单次使用space中定义的A。::为域解析符

using space::A：导入space中定义的A

**导入库**

#include<A.h>：导入头文件A.h。A.h中存放源文件A.cpp中的函数声明，相当于导入A库，系统将从系统目录开始搜寻A.h文件

#include”A.h”：系统先从当前目录搜寻A.h文件，再从系统目录搜寻

#include<iostream>：导入C++必要库iostream(不加.h)，包含标准名字空间std。对于一般的库，加或不加.h都可以

using namespace std;：使用std名字空间定义的函数

**数据定义**

int a,b;/int a=1,b=2;：定义整型(int32)变量。如果赋值a=1.1，会自动把1.1转化为int型

float a=1.1;：定义float(float32)型变量。如果赋值a=1，会自动把1转换为float32型

double a=1.1;：定义double(float64)型变量

bool a=1;：定义bool类型。a的值只有0/1，会自动转换其他值为0/1

const int/float/double/bool a=1;：定义常量。必须在定义时赋值

constexpr int a=1：编译时常量。在编译时会直接将a替换为1，性能比const高，但由于在编译时优化，不能包含函数调用和对象构造

**引用**

(const)int/float/double &b=a;：b为a的引用，共用内存，会一同改变。a为已经定义好的数据，b定义时要与a的类型一致，如果不一致在某些情况下会单独创造内存而不是引用。不能对数组引用

void 函数名(int x,int &y,const int &z){...}：int x参数传入后会创造一个内存空间；int &y公用内存，函数中参数改变时被引用的参数也会改变；const int &z可以在节约内存时更保险的防止参数被改变。参数的引用定义要与主函数中的定义一致，否则会单独创造内存而不是引用

**指针**

指针占用8bit，是专门存放内存地址的变量，可以向系统申请动态内存空间，获得某段已有内存的首地址。使用步骤：定义、指向已有地址/申请内存并赋值、释放。

对于\*p：&p为指针的地址，p为指针的值(指向的地址)，p+a为p指向地址的相邻a个地址，如果是int32型，相邻地址间相差4字节。\*p/p[0]为地址p的值，p[a]为地址p+a的值。对于\*\*p，

由于指针指向的是地址，即使p[1]、p[2]等没有申请也可以查看和操作

int \*p;：定义一级指针

p=&A;：使指针指向A的地址，改变指向前要注意释放之前申请的内存

p=new int;：申请一个新的int内存

delete p;：释放p所指的内存

p=new int[a];：申请连续a个新的int内存

delete []p;：释放p所指的连续内存。！如果不加[]会造成内存占用！

\*p=b;：给地址p赋值。p[a]=b;：给地址p+a赋值

int \*\*p;：定义二级指针

p=new int\*[a];for(i=0;i<c;i++)//p[i]=new int[b];：创建a×b指针

for(i=0;i<c;i++)//delete []p[i];//delete []p;：释放所有空间

**函数**

函数需先定义后调用，如果函数写在调用后，需要在调用前加函数声明

void 函数名(...)：函数没有返回值

int(/...) 函数名(...)：函数返回规定类型的值，但可以不接受

int(/...) & 函数名(int &x){return x}：引用函数。函数返回的值为主函数中x的引用。常见为左值调用：函数名(...)=1

引用类型：

类型(int/bool) 函数名(类型 A,...){...}定义函数。

类型(int/bool) 函数名(类型 A,...);：函数声明

**条件语句**

if 条件 end(or) 条件://...//elif ...://...//else://...

for(i=0;i<a;i++)//...;：循环，i需提前定义

a++：自加1

a--：自减1

**逻辑运算符**

&&：与

||：或

!：异或

**位运算符**

位运算符优先级较低

a<<n：二进制左位移n次。等于乘以2的n次方，运算效率更高

a>>n：二进制右位移n次。等于除以2的n次方，运算效率更高

&：二进制与

|：二进制或

^：二进制异或

~：二进制非

**其他运算符**

sizeof(a)：返回a的占位大小

function?x:y：function为True执行x，否则执行y

A.a：返回结构体对象A的内容a

P->p：返回结构体指针P的内容p

&a：a的地址

\*a：指针a对应的值

**《c++常用库》**

**<stdio.h>**

c语言的标准输入输出库，有printf、scanf,、puts,、gets等函数，c++也可以使用，但不支持面向对象编程、不提供类型安全。c++有自己的<iostream>库，提供更丰富的功能

**<iostream>**

c++的标准输入输出库，早期版本使用<iostream.h>

**<string>**

处理字符串的库，不用关心内存分配和\0

string str1;：定义空字符串

getline(cin,str1);：输出带空格的字符串

str3=str1+“ “+str2;——字符串合并，中间可加空格

str1.insert(a,”A”);——在str1下标为a的字符前插入字符A

str1.replace(a,b,”A”);——从下标a开始往后共b个字符替换为A

str1.substr(a,b);——选取下标a开始往后共b个字符

str1.erase(a,b);——删除下标a开始往后共b个字符

str1.find(“A”);——查找str1中是否含有A，存在返回A第一个字符的下标，不存在返回-1

**<eigen>**

安装教程：<https://blog.csdn.net/qq_45948208/article/details/132434560>

**<opencv2>**

(下载安装包，将相关变量添加到系统变量和VS目录、链路器中)

下载网址：https://opencv.org/releases/

using namespace cv;：使用cv库

**tenorrt**

C++深度学习模型加速

