Part 1 基础知识

--Zhen Wang

能Google到的就不要参考我的Slides！！！

目录

- 复习操作系统的基础知识
- 学习C/C++基本操作
- 学习高能物理常用数据分析软件ROOT
- 进入到下一阶段的课程

先来看点大家熟悉的东西

操作系统

应用程序

语言处理系统

操作系统

指令集体系结构

计算机硬件

解决用户需求
各类程序

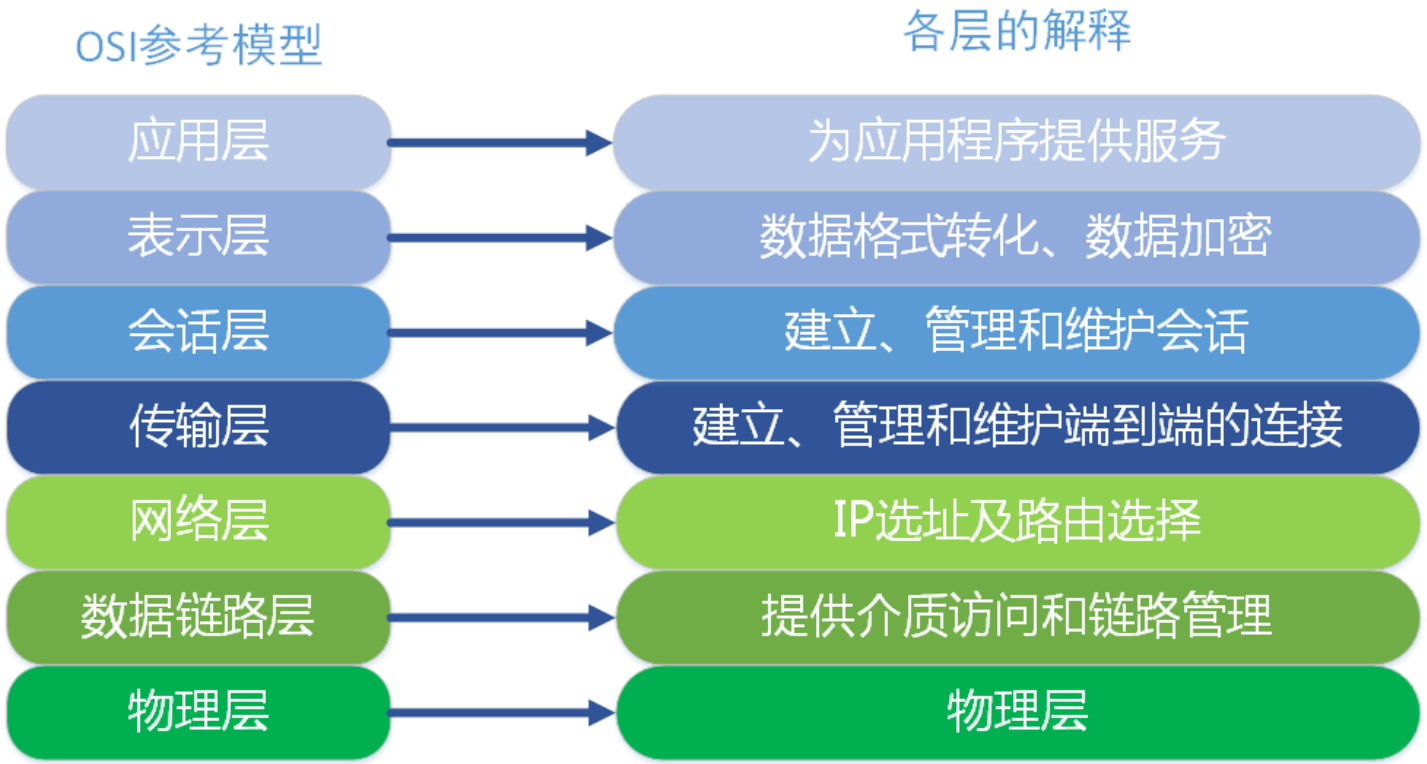
各种语言处理程序
(编译、汇编、链接)
运行时系统
(库函数、调试、优化等)

对内管理员 对外服务员
资源管理者和控制者

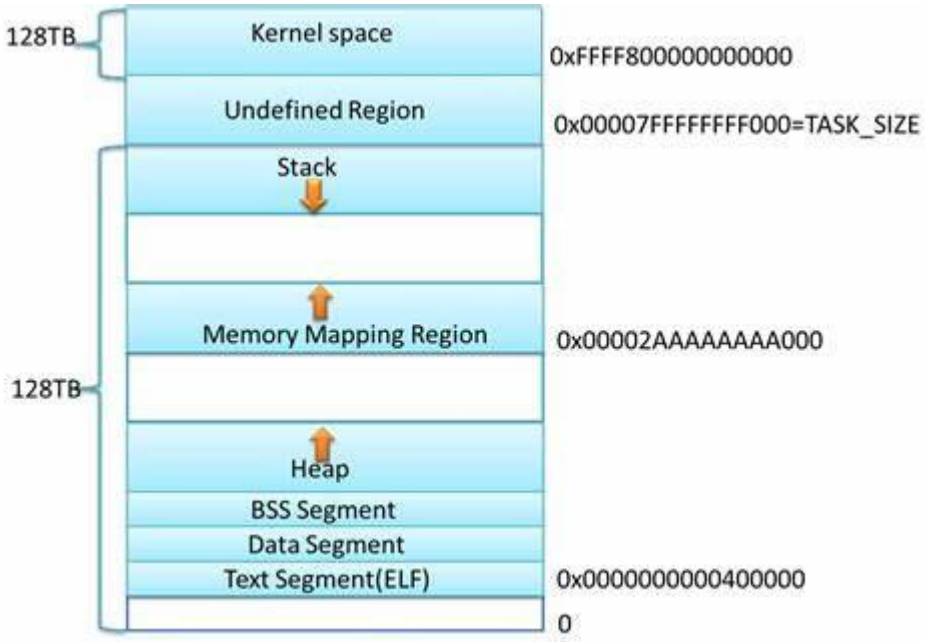
软件硬件的界面
对硬件的抽象

执行与实施

复习基础知识



程序的内存模型



复习基础知识


OSI参考模型
应用层
表示层
会话层
传输层
网络层
数据链路层
物理层
序的内存模型
Kernel space
0xFFFF800000000000
Undefined Region
0x00007FFFFFFFF000=TASK_SIZE
Stack
Memory Mapping Region
0x00002AAAAAAAA000
Heap
BSS Segment
Data Segment
Segment(ELF)
0x0000000000400000
0

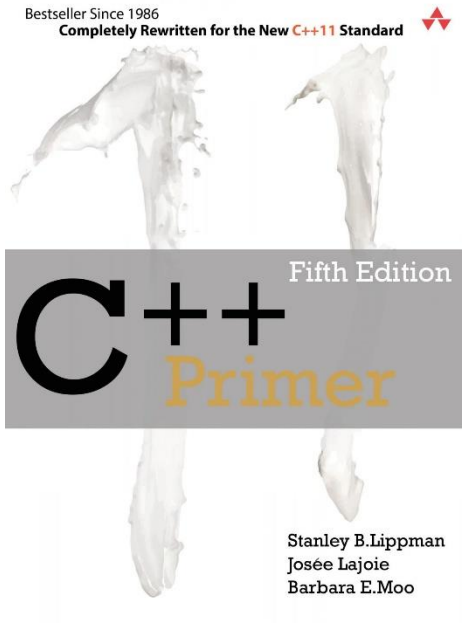
什么是我们（不）应该做的?

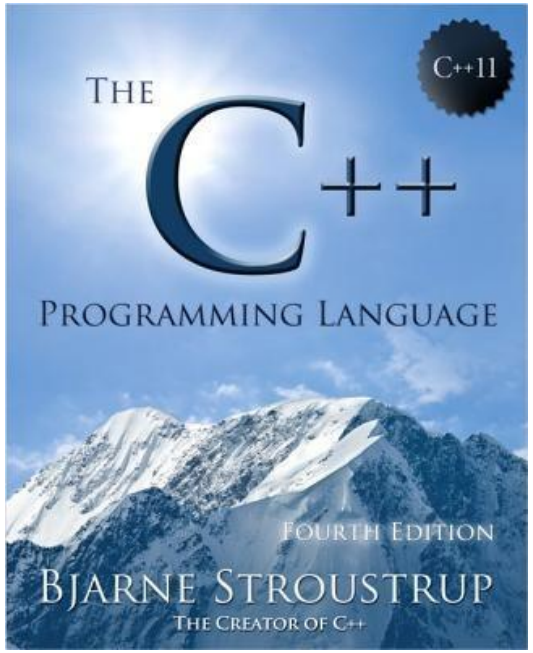
- **不应该**:
  - 卖弄学识!
  - 快速地讲一堆名词，谁也听不懂!
  - 学生听完满头雾水!
- **应该**:
  - 结果导向!
  - 学生会写C++了!
  - 学生会由内而外不断产生动力!

比起记名词，应该先学会用!

引用一句: “就算是一头猪从头听到尾也会了!!”

什么是我们可以做的?





- 可以:
  - 完全复制粘贴别人的PPT!
  - 教科书般地解释每一个名词!!
  - 一字不落地念完PPT上每一个字!!!
- 结论:
  - 尽可能让每个人从什么都不懂，到能独立写出一个项目!
- 作为学生:
  - 不会就举手问!
- Reference:
  - 菜鸟教程
  - cppreference.com
  - ChatGPT
  - Google (Stack overflow)

为什么我们需要学习C++?

WLCG网格站点

熟练掌握C++能够帮助想法的快速落地!

加速物理分析的进程!

ROOT - C++ 环境配置

- 登录bl0 (让我知道如果你不会)
- 键入在你的终端:
  - source /cvmfs/sft.cern.ch/lcg/views/LCG_102/x86_64-centos7-gcc11-opt/setup.sh
  - which root (你正在使用lcg102里的ROOT)
  - 创建一个文件夹作为你的临时工作目录 (mkdir test)

```
[Azure ☭ bl-0 10:57 PM]~[~]
 --->>>  which root
/cvmfs/sft.cern.ch/lcg/views/LCG_102/x86
_64-centos7-gcc11-opt/bin/root
```

```
root [0] std::cout<<"Hello World!"<<std::endl;
Hello World!
root [1]
```

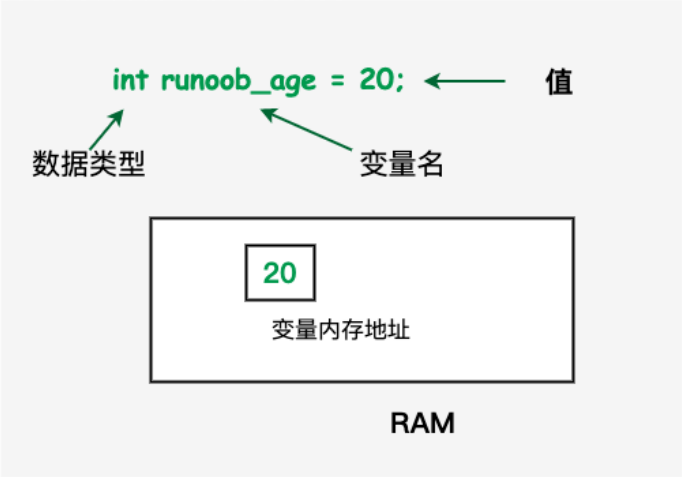
```
[Azure ☭ bl-0 10:58 PM]~[~]
 --->>>  root
  ------------------------------------------------------------------
  | Welcome to ROOT 6.26/04                        https://root.cern |
  | (c) 1995-2021, The ROOT Team; conception: R. Brun, F. Rademakers |
  | Built for linuxx8664gcc on Jun 07 2022, 16:01:16                 |
  | From tags/v6-26-04@v6-26-04                                      |
  | With g++ (GCC) 11.2.0                                            |
  | Try '.help', '.demo', '.license', '.credits', '.quit'/'.q'       |
  ------------------------------------------------------------------

root [0]
```

C++ 基本数据类型

数据类型	字节数	最小值	最大值
bool	1	0	1
char	1	~	~
int	4	-2147483648	2147483647
unsigned int	4	0	4294967295
short int	2	-32768	32767
long int	8	-9.2234e18	9.2234e18
float	4	-1.1754e38	3.4028e38
double	8	-2.2250e308	1.7976e308

- 键入在你的终端：
  - root (打开root)
  - int a = 1;
  - cout<<a<<endl;

```
[Azure ☭ bl-0 11:07 PM]~[~]
 ---->>>  root -l
root [0] int a = 1;
root [1] cout<<a<<endl;
1
root [2]
```



你定义了一个int类型的变量a！并且给它赋了初值为1!

C++ 常量

- const 前缀加到类型前面

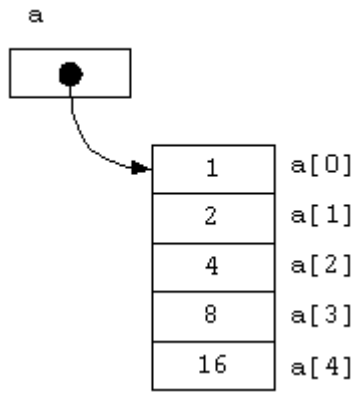
- 在你的root终端输入:
  - const int a = 1;
  - a = 2 ;

```
[Azure ☭ bl-0 11:23 PM]~[~]
 --->>>  root -l
root [0] const int a = 1;
root [1] a=2
ROOT_prompt_1:1:2: error: cannot assign to variable 'a' with const-qualified type 'co
nst int'
a=2
~^
ROOT_prompt_0:1:11: note: variable 'a' declared const here
const int a = 1;
~~~~~~~~~~^~~~
root [2]
```

在分析当中可以发挥什么用处?

数组

- 一个固定大小、相同类型元素的顺序集合



- int a[3] = {1,2,3};
- cout<<a[0]<<endl;

```
[Azure ☭ bl-0 11:36 PM]~[~]
 --->>>  root -l
root [0] int a[3]={1,2,3};
root [1] cout<<a[0]<<endl;
1
root [2]
```

数组是很多数据结构的底层实现方法，之后会经常遇到!

在分析当中可以发挥什么用处?

C/C++ 字符串

- char name[] = “DarkShine”;
- cout<<name[0]<<endl;
- cout<<name<<endl;

- std::string name2 = “DarkShine2”;
- cout<<name2[0]<<endl;
- cout<<name2<<endl;

在分析当中可以发挥什么用处?

循环&&条件判断

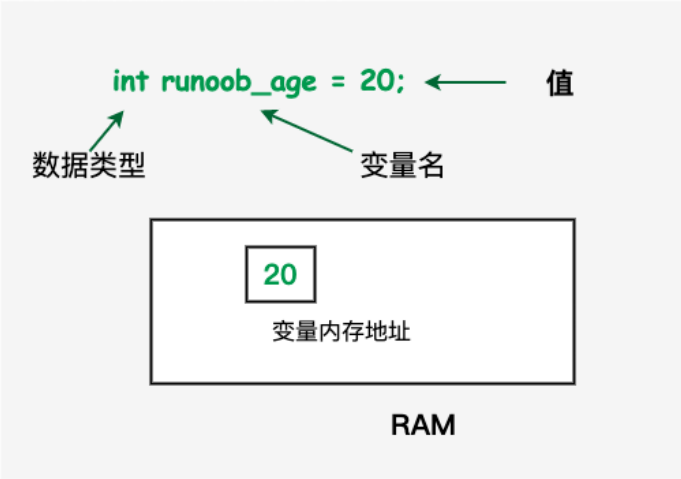
- for(int i=0;i<20;i++){
- cout<<i<<“\n”<<endl;
- if(i>10)break;
- }
- while(1){
- // ..
- }

在分析当中可以发挥什么用处?

指针

- int a = 1;
- cout << &a << endl;

- //如何声明?
- int *b;
- //如何赋值
- b = &a;
- //如何查看指向的值
- cout << *b << endl;



C++ 引用

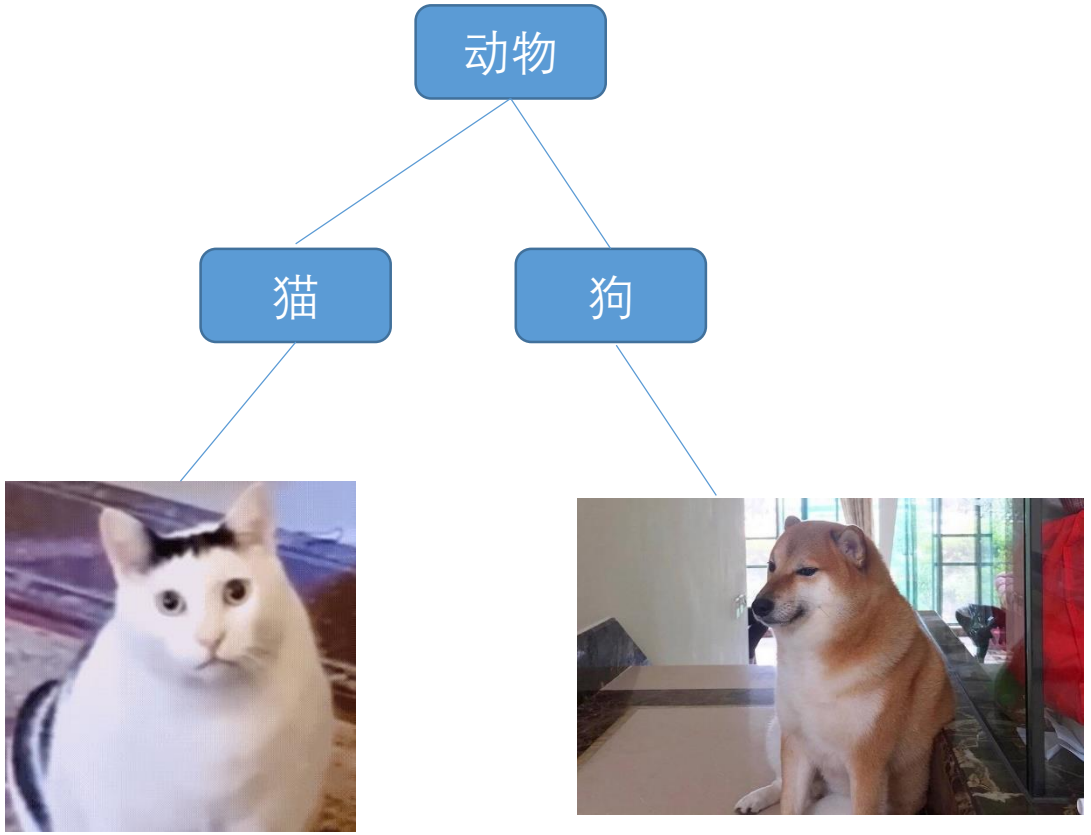
引用一句: “就算是一头猪从头听到尾也会了!!”

- 我们今天已经用过引用了
- string gword = “Even a pig can learn !”;
- string& zhenword = gword;
- // 说的是同一个东西
- zhenword = “But I can not!”
- cout<< gword << endl;

- 不存在空引用，既然是引用必须意有所指

C++ 类

面向对象程序设计的核心



bruh

cheems

类大体包括了成员变量和成员函数

通过一个类可以定义出来多个同类型的对象

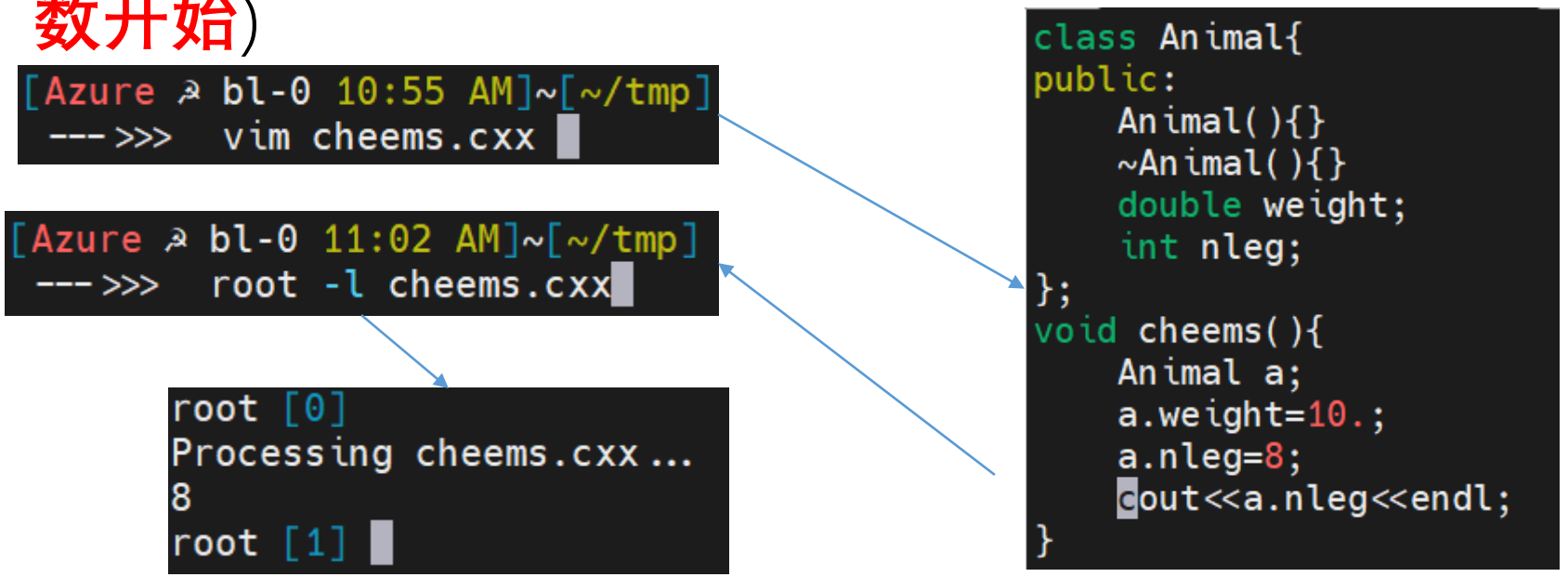
如何定义类

```
class Animal{
public:
  Animal(){} //构造函数 constructor
  ~Animal(){} //析构函数 destructor
  double weight;
  int nleg;
};
```

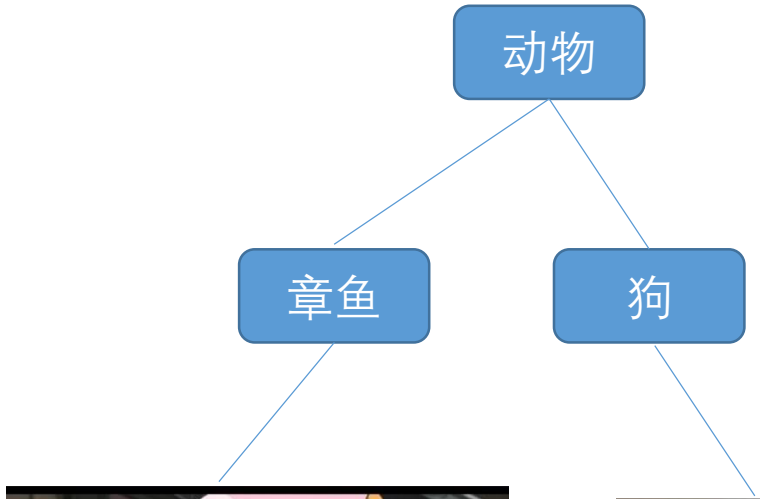
```
// 如何创建对象
 Animal a;
 a.weight=10.;
 a.nleg=8;

 cout<<a.weight<<endl;
```

把代码保存下来

- 我们发现每次都重新敲那些代码已经变成了一种无意义的重复劳动
- ROOT 支持在线编译执行你写好的脚本
- 比如编写了一个叫做cheems.cxx的脚本 (**主程序一般都从main函数开始**)



C++继承



我们已经创建出了动物，但是cheems不仅有体重和腿，它还有颜色，还会发出汪汪汪的叫声

我们要创建出来狗的类，他必须有四条腿!

```
class Animal{
public:
    Animal(){}
    ~Animal(){}
    void eat(){}
    void sleep(){}
    void play(){}
    double weight;
    int nleg;
};
class Dog : public Animal{
public:
    Dog(){}
    ~Dog(){}
    const int nleg=4;
};
```

```
void cheems(){
    Dog d;
    cout<<d.nleg<<endl;
}
```

```
root [0]
Processing cheems.cxx...
4
root [1]
```

C++类数据 抽象&封装

C++ 数据抽象

数据抽象是指，只向外界提供关键信息，并隐藏其后台的实现细节，即只表现必要的信息而不呈现细节。

C++ 数据封装

所有的 C++ 程序都有以下两个基本要素：

- **程序语句（代码）：**这是程序中执行动作的部分，它们被称为函数。
- **程序数据：**数据是程序的信息，会受到程序函数的影响。

封装是面向对象编程中的把数据和操作数据的函数绑定在一起的一个概念，这样能避免受到外界的干扰和误用，从而确保了安全。

复制这个文件夹到你的工作目录下:
/2024-winter-software-tutorial/zhen_material/cheems

然后解释一下这个小程序(到此为止里面的东西都是你已经掌握的)

你真棒!

C++多态

C++ 多态

多态按字面的意思就是多种形态。

```cpp
#include<iostream>

using namespace std;

class Animal{
public:
    void takeShower(){cout<<"Animal taking shower"<<endl;}
    void eat(){cout<<"Animal eating"<<endl;}
};

class Dog : public Animal{
public:
    void takeShower(){cout<<"Dog taking shower"<<endl;}
    void eat(){cout<<"Dog eating"<<endl;}
};

int main(){
    Animal *a;
    Dog d;
    a=&d;
    a->eat();
    return 1;
}
```

基类的eat函数

派生类的eat函数

基类指针

派生类对象

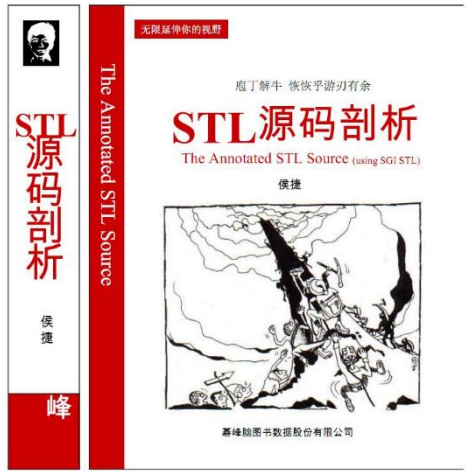
基类指针指向了派生类对象

基类指针调用eat函数

一点点STL

- STL(Standard Template Library) 是C++标准库的一部分，每次使用的时候只需要include头文件就可以
- STL有几大组件，其中我们经常打交道的就是一些容器、算法和迭代器

- 主要介绍vector, map这两个容器，剩下的请自行学习



std::vector

- #include<vector>
- ...
- vector<int> v; //创建一个vector<int> 此时长度为0 v.size()
- v.push_back(1); //推入一个元素
- v.push_back(4); //再推入一个元素
- v.size(); //此时为2
- v.at(0); //v[0] 也可以访问到某个元素
- //自行查询如何遍历vector的元素

std::map

- #include<map> //map 是key-value的pair的容器
- …
- map<string,int> m;
- m[“Zhen”]=18;
- m[“Yulei”]=8;
- m[“Xuliang”]=7;
- cout<<m[“Xuliang”]<<endl;
- cout<<boolalpha<<(m[“Zhen”]>m[“Yulei”])<<endl;
- //自行查询和学习遍历map的操作，如果你在遍历的时候发现了一些规律，思考一下为什么是这样的

ROOT

- Reference :
  - https://root.cern/doc/master/

- ROOT是高能物理内广泛使用的数据分析**工具**

- 工具是服务于物理的

ROOT File

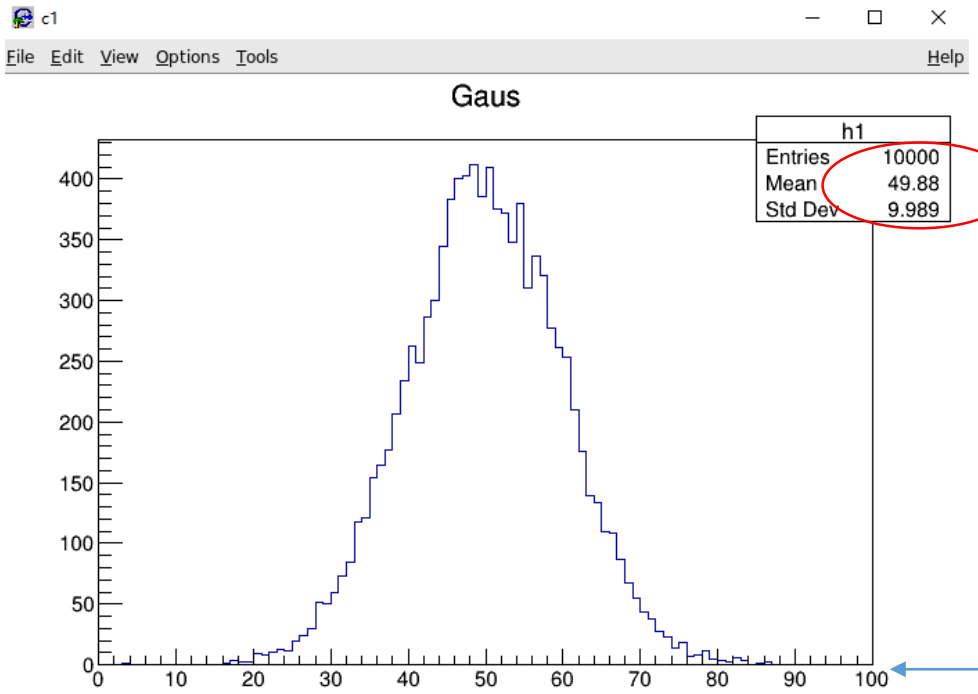
- 复制这个文件到你的工作目录下
- /home/wangzhen/tmp/ROOTFILE/result.root
- 打开这个文件 (root –l result.root)
- 查看这个文件里面的内容(.ls)

```
 --->>>  root -l result.root
root [0]
Attaching file result.root as _file0...
(TFile *) 0x2cfd060
root [1] .ls
TFile**         result.root
 TFile*         result.root
  KEY: TH1D     h1;1    Gaus
  KEY: TTree    tree;1  Gaus tree
root [2]
```

ROOT File

- 如何查看这个TH1D?
- h1->Draw()

```
 --->>>  root -l result.root
root [0]
Attaching file result.root as _file0...
(TFile *) 0x2cfd060
root [1] .ls
TFile**         result.root
 TFile*         result.root
  KEY: TH1D     h1;1    Gaus
  KEY: TTree    tree;1  Gaus tree
root [2]
```



信息统计

这个直方图里面一共填了10000个数
这些数字呈现出来的分布有个均值49.88
这些数字的标准差是9.989

多少个bin，x轴范围是从哪到哪

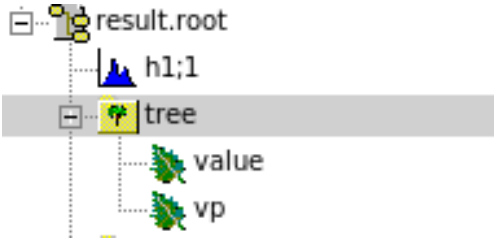
ROOT File

- 如何查看这个TTree?
- tree->Print()
- tree->Show(0)
- tree->GetEntries()//一共多少个条目

现在你要问了 什么是TTree?
new TBrowser

```
root [4] new TBrowser
(TBrowser *) 0x2d33520
root [5]
```

```
root [3] tree->Print()
******************************************************************************
*Tree    :tree      : Gaus tree                                              *
*Entries :    10000 : Total =          161749 bytes  File  Size =      83886 *
*        :          : Tree compression factor =   1.92                       *
******************************************************************************
*Br    0 :value     : value/D                                                *
*Entries :    10000 : Total  Size=      80709 bytes  File Size  =      71143 *
*Baskets :        3 : Basket Size=      32000 bytes  Compression=   1.13     *
*.............................................................................*
*Br    1 :vp        : vp/D                                                   *
*Entries :    10000 : Total  Size=      80688 bytes  File Size  =      12217 *
*Baskets :        3 : Basket Size=      32000 bytes  Compression=   6.57     *
*.............................................................................*
```



关于查找Reference

- 你发现你想用某个类，但是你不知道如何使用它怎么办？
- 答案是：上网查！查什么？Google？ChatGPT？
- 此处现场示范，没有内容

- Little project 在代码里创建一个TH1D并且让它的Title为你的名字，10个bin，范围从0到10，并且随便Fill几个数进去，在代码的最后把它画出来

Project A&B

- TH1D *h=new TH1D(“h”,”Zhen”,10,0,10);
- h->Fill(5);
- h->Draw();

- //Project B 创建一个TFile 并且把这个TH1D存进去
- TFile *fout = new TFile(“result_B.root”,”RECREATE”);
- // RECREATE 这个option代表你要新建这个文件
- fout->cd(); //进入这个文件
- h->Write(); //把hist 写到文件里
- fout->Close(); //关闭文件

如何创建TTree？Project C

- TTree *t = new TTree(“中间应该写什么？”);
- int i ; //声明一个本地变量作为Branch的载体
- t->Branch(“br_name[什么名字都可以]”,&i); //把i绑定到这个tree的branch上面
- for(int j=0;j<100;j++){
- i=j;
- t->Fill(); // 填一次这个tree / 给tree增加一个Entry
- }
- //新建一个文件，并且把这个tree写到这个新文件里面。

如何读取TTree里面的数据

- TFile *f = TFile::Open(“你的ROOTFILE.root”,”READ”);
- TTree *t = (TTree*)f->Get(“Tree的名字”); //C风格的强制类型转换
- //为什么这里需要强制类型转换?
- //C++风格的强制类型转换又是什么样的?
- int br; //生命一个本地变量来存储你的branch
- t->SetBranchAddress(“br_name [这里就得按照实际branch的名字来填写了] ”,&br); //把本地的br变量和tree的branch映射起来
- t->GetEntry(0); //这时候tree就进入到了第1个entry里面
- cout<<br<<endl;

Final Project

- 读取这个ROOT文件中的ECAL_E_total变量，并且将这个vector里面的第一个元素的分布画出来。
- /lustre/collider/wangzhen/tmp/cpp_class/dp_ana.root

- Google,ChatGPT,各种搜索引擎都是你的好伙伴!

Summary

- 如果你画出了这个分布，那就说明你已经初步具备了分析DarkShine输出的文件的能力！
- 恭喜你！第一阶段课程的目的达到了！