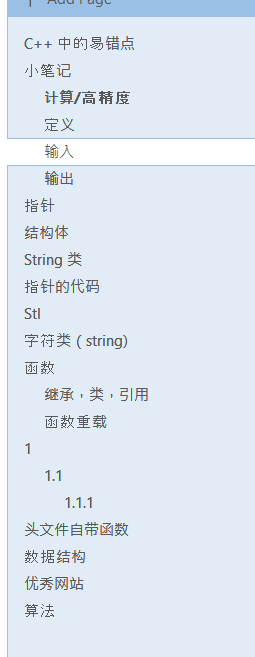
# C语言学习报告

在c语言的学习过程中，我愈加清晰的了解到记笔记的重要性，无论是语法的使用，还是算法的优化和比较，以至于错误的总结和整理，总之，在新手的学习阶段，笔记是一个非常重要的过程。其重要意义在于，在自己去整理的过程中，我们看到了自己的不足，在下次的训练和练习时，就能够自动的看到自己的不组，以此，来实现自动的更正，以下为我的笔记节选

* 易错：

1. ;和中文的；分不清楚，导致出错

* D:\code\C++> cd "d:\code\C++\" ; if ($?) { g++ what.cpp -o what } ; if ($?) { .\what }此时， 电脑已经开始运行，就可以进行试验了

for(int i=1;i<=a;i++)

    {

        cin>>\*p;//    在这里每次的输入都很正常

        p++;//但是，到这里，每次都会再加上，导致最后的p指向的是p【3】，不存在

    }

注意，其中的p最终指向的是谁！

1. 注意，使用了new 唤醒了指针，一定要再加上delete用于维护系统，否则可能造成很大的问题！
2. p[i]无法使用？？是的，好像赋值都没法实现
3. Struct 结构体的最后需要加上;
4. ld return 1 exit status指的是

**1、库函数拼写错误。如：printf，scanf,等**

**2、定义的函数名在调用时，函数名拼写错误**

**3、在执行函数时可能函数体本身的问题**

**4、关于其他人说的有程序在运行，这一点我倒是没发现（可能是我还没遇到）**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **已知！要早于&&，因此，优先级越大，有靠后执行** | | | | |  |
| **优先级** | **运算符** | **描述** | **结合性** |
| **1** | :: | [作用域解析](https://zh.cppreference.com/w/cpp/language/identifiers#.E6.9C.89.E9.99.90.E5.AE.9A.E7.9A.84.E6.A0.87.E8.AF.86.E7.AC.A6) | 从左到右 |
| **2** | a++   a-- | 后缀[自增与自减](https://zh.cppreference.com/w/cpp/language/operator_incdec) |  |
|  | *type*()   *type*{} | [函数风格转型](https://zh.cppreference.com/w/cpp/language/explicit_cast) |  |
|  | a() | [函数调用](https://zh.cppreference.com/w/cpp/language/operator_other#.E5.86.85.E5.BB.BA.E7.9A.84.E5.87.BD.E6.95.B0.E8.B0.83.E7.94.A8.E8.BF.90.E7.AE.97.E7.AC.A6) |  |
|  | a[] | [下标](https://zh.cppreference.com/w/cpp/language/operator_member_access#.E5.86.85.E5.BB.BA.E7.9A.84.E4.B8.8B.E6.A0.87.E8.BF.90.E7.AE.97.E7.AC.A6) |  |
|  | .   -> | [成员访问](https://zh.cppreference.com/w/cpp/language/operator_member_access#.E5.86.85.E5.BB.BA.E7.9A.84.E6.88.90.E5.91.98.E8.AE.BF.E9.97.AE.E8.BF.90.E7.AE.97.E7.AC.A6) |  |
| **3** | ++a   --a | 前缀[自增与自减](https://zh.cppreference.com/w/cpp/language/operator_incdec) | 从右到左 |
|  | +a   -a | 一元[加与减](https://zh.cppreference.com/w/cpp/language/operator_arithmetic" \l ".E4.B8.80.E5.85.83.E7.AE.97.E6.9C.AF.E8.BF.90.E7.AE.97.E7.AC.A6) |  |
|  | !   ~ | [逻辑非](https://zh.cppreference.com/w/cpp/language/operator_logical)和[逐位非](https://zh.cppreference.com/w/cpp/language/operator_arithmetic#.E6.8C.89.E4.BD.8D.E9.80.BB.E8.BE.91.E8.BF.90.E7.AE.97.E7.AC.A6) |  |
|  | (*type*) | [C 风格转型](https://zh.cppreference.com/w/cpp/language/explicit_cast) |  |
|  | \*a | [间接](https://zh.cppreference.com/w/cpp/language/operator_member_access#.E5.86.85.E5.BB.BA.E7.9A.84.E9.97.B4.E6.8E.A5.E8.BF.90.E7.AE.97.E7.AC.A6)（解引用） |  |
|  | &a | [取址](https://zh.cppreference.com/w/cpp/language/operator_member_access#.E5.86.85.E5.BB.BA.E7.9A.84.E5.8F.96.E5.9C.B0.E5.9D.80.E8.BF.90.E7.AE.97.E7.AC.A6) |  |
|  | sizeof | [取大小](https://zh.cppreference.com/w/cpp/language/sizeof)[[注 1]](https://zh.cppreference.com/w/cpp/language/operator_precedence#cite_note-1) |  |
|  | co\_await | await 表达式 (C++20) |  |
|  | new   new[] | [动态内存分配](https://zh.cppreference.com/w/cpp/language/new) |  |
|  | delete   delete[] | [动态内存分配](https://zh.cppreference.com/w/cpp/language/delete) |  |
| **4** | .\*   ->\* | [成员指针](https://zh.cppreference.com/w/cpp/language/operator_member_access#.E5.86.85.E5.BB.BA.E7.9A.84.E6.88.90.E5.91.98.E6.8C.87.E9.92.88.E8.AE.BF.E9.97.AE.E8.BF.90.E7.AE.97.E7.AC.A6) | 从左到右 |
| **5** | a\*b   a/b   a%b | [乘法、除法与余数](https://zh.cppreference.com/w/cpp/language/operator_arithmetic#.E4.B9.98.E6.B3.95.E6.80.A7.E8.BF.90.E7.AE.97.E7.AC.A6) |  |
| **6** | a+b   a-b | [加法与减法](https://zh.cppreference.com/w/cpp/language/operator_arithmetic#.E5.8A.A0.E6.B3.95.E6.80.A7.E8.BF.90.E7.AE.97.E7.AC.A6) |  |
| **7** | <<   >> | 逐位[左移与右移](https://zh.cppreference.com/w/cpp/language/operator_arithmetic" \l ".E7.A7.BB.E4.BD.8D.E8.BF.90.E7.AE.97.E7.AC.A6) |  |
| **8** | <=> | [三路比较运算符](https://zh.cppreference.com/w/cpp/language/operator_comparison#.E4.B8.89.E8.B7.AF.E6.AF.94.E8.BE.83)(C++20 起) |  |
| **9** | <   <= | 分别为 < 与 ≤ 的[关系运算符](https://zh.cppreference.com/w/cpp/language/operator_comparison) |  |
|  | >   >= | 分别为 > 与 ≥ 的[关系运算符](https://zh.cppreference.com/w/cpp/language/operator_comparison) |  |
| **10** | ==   != | 分别为 = 与 ≠ 的[关系运算符](https://zh.cppreference.com/w/cpp/language/operator_comparison) |  |
| **11** | a&b | [逐位与](https://zh.cppreference.com/w/cpp/language/operator_arithmetic#.E6.8C.89.E4.BD.8D.E9.80.BB.E8.BE.91.E8.BF.90.E7.AE.97.E7.AC.A6) |  |
| **12** | ^ | [逐位异或](https://zh.cppreference.com/w/cpp/language/operator_arithmetic#.E6.8C.89.E4.BD.8D.E9.80.BB.E8.BE.91.E8.BF.90.E7.AE.97.E7.AC.A6)（互斥或） |  |
| **13** | | | [逐位或](https://zh.cppreference.com/w/cpp/language/operator_arithmetic#.E6.8C.89.E4.BD.8D.E9.80.BB.E8.BE.91.E8.BF.90.E7.AE.97.E7.AC.A6)（可兼或） |  |
| **14** | && | [逻辑与](https://zh.cppreference.com/w/cpp/language/operator_logical) |  |
| **15** | || | [逻辑或](https://zh.cppreference.com/w/cpp/language/operator_logical) |  |
| **16** | a?b:c | [三元条件](https://zh.cppreference.com/w/cpp/language/operator_other#.E6.9D.A1.E4.BB.B6.E8.BF.90.E7.AE.97.E7.AC.A6)[[注 2]](https://zh.cppreference.com/w/cpp/language/operator_precedence#cite_note-2) | 从右到左 |
|  | throw | [throw 运算符](https://zh.cppreference.com/w/cpp/language/throw) |  |
|  | co\_yield | yield 表达式 (C++20) |  |
|  | = | [直接赋值](https://zh.cppreference.com/w/cpp/language/operator_assignment#.E5.86.85.E5.BB.BA.E7.9A.84.E7.9B.B4.E6.8E.A5.E8.B5.8B.E5.80.BC)（C++ 类默认提供） |  |
|  | +=   -= | 以和及差[复合赋值](https://zh.cppreference.com/w/cpp/language/operator_assignment" \l ".E5.86.85.E5.BB.BA.E7.9A.84.E5.A4.8D.E5.90.88.E8.B5.8B.E5.80.BC) |  |
|  | \*=   /=   %= | 以积、商及余数[复合赋值](https://zh.cppreference.com/w/cpp/language/operator_assignment" \l ".E5.86.85.E5.BB.BA.E7.9A.84.E5.A4.8D.E5.90.88.E8.B5.8B.E5.80.BC) |  |
|  | <<=   >>= | 以逐位左移及右移[复合赋值](https://zh.cppreference.com/w/cpp/language/operator_assignment" \l ".E5.86.85.E5.BB.BA.E7.9A.84.E5.A4.8D.E5.90.88.E8.B5.8B.E5.80.BC) |  |
|  | &=   ^=   |= | 以逐位与、异或及或[复合赋值](https://zh.cppreference.com/w/cpp/language/operator_assignment" \l ".E5.86.85.E5.BB.BA.E7.9A.84.E5.A4.8D.E5.90.88.E8.B5.8B.E5.80.BC) |  |
| **17** | , | [逗号](https://zh.cppreference.com/w/cpp/language/operator_other#.E5.86.85.E5.BB.BA.E7.9A.84.E9.80.97.E5.8F.B7.E8.BF.90.E7.AE.97.E7.AC.A6) | 从左到右 |

1.指针的动态联编很好用，可以用于解决一些静态联编无法解决的问题，另一种实现方式位VECTOR!

2.指针下的结构体的两种调用方式：

1.ps->name 2.(\*ps).price

3.A++是先运行再计算 ++a是先计算再运行

4.for 循环的中间部分属于判断执行的条件，在此期间，需要注意，当判断结果为FALSE 时，循环就会停止！！！！而false=0

5.

|  |  |  |
| --- | --- | --- |
| 属性 | Cin.get(ch) | Ch=cin.get() |
| 传递字符的方式 | 赋值型 | 返回型（在这个过程中，cin.get()被当作是函数，返回了一个char型的值） |
| 输入的字符 | True | int型的编码 |
| EOF时的返回值 | false | EOF |

1. 对于“==”的使用 建议使用“value==variable”：

Example: 3==a 如果写成了 3=a 会更加简单的识别

相反，如果写成了 a=3 很难察觉

1. 在判断语句中不能够使用a<b<c eg:17<A<35==(17<A)<35 相当于是一个判断式，结果为1/0 再与35进行比较，分清数学和计科
2. INT\_MAX&INT\_MIN 二者都在头文件 limits 中。
3. 在while 循环中使用continue是一种非常可怕的事情，因为在continue的作用下，循环会自动跳过后边的内容，又因为while循环的累加是在最后完成的，所以会陷入死循环！

不过，在for循环中一般不会出现这种问题。。因为for的累加是在开头完成的！Whil

1. 在程序的最后如果出现了cin.get()则是类似system（“pause”）的作用

|  |  |  |
| --- | --- | --- |
| 转义字符 | 意义 | ASCII码值（十进制） |
| \a | 响铃(BEL) | 007 |
| \b | 退格(BS) ，将当前位置移到前一列 | 008 |
| \f | 换页(FF)，将当前位置移到下页开头 | 012 |
| \n | 换行(LF) ，将当前位置移到下一行开头 | 010 |
| \r | 回车(CR) ，将当前位置移到本行开头 | 013 |
| \t | 水平制表(HT) （跳到下一个TAB位置） | 009 |
| \v | 垂直制表(VT) | 011 |
| \\ | 代表一个反斜线字符''\' | 092 |
| \' | 代表一个单引号（撇号）字符 | 039 |
| \" | 代表一个双引号字符 | 034 |
| \? | 代表一个问号 | 063 |
| \0 | 空字符(NUL) | 000 |
| \ddd | 1到3位八进制数所代表的任意字符 | 三位八进制 |
| \xhh | 十六进制所代表的任意字符 | 十六进制 |

*From <*[*https://baike.baidu.com/item/%E8%BD%AC%E4%B9%89%E5%AD%97%E7%AC%A6/86397?fr=aladdin*](https://baike.baidu.com/item/%E8%BD%AC%E4%B9%89%E5%AD%97%E7%AC%A6/86397?fr=aladdin)*>*

其中，\r的意思是将光标回到本行的首位

而\n则是将光标运行到下行的首位

* 实战过程中的一些总结
* **c++ 模块**

# 新内容

* 1.string的vector存储：
* 首先通过一个char数组，读取然后再通过insert输入到文件vector中存储
* 2.文件的读写
* A.这里采用的是fopen:
* FILE \*p=fopen("filename","mode")
* 其中，mode 有

|  |  |
| --- | --- |
| **模式** | **描述** |
| "r" | 打开一个用于读取的文件。该文件必须存在。 |
| "w" | 创建一个用于写入的空文件。如果文件名称与已存在的文件相同，则会删除已有文件的内容，文件被视为一个新的空文件。 |
| "a" | 追加到一个文件。写操作向文件末尾追加数据。如果文件不存在，则创建文件。 |
| "r+" | 打开一个用于更新的文件，可读取也可写入。该文件必须存在。 |
| "w+" | 创建一个用于读写的空文件。 |
| "a+" | 打开一个用于读取和追加的文件。 |

* 注意区分w/a
* 在文件读取时，采用的是fscanf/fprintf
* 3.EOF的应用：
* 在文件中EOF是文件的最后标志，以此可以结束输入
* 采用的是EOF!=fscanf(p(在2中提及的指针),"%s",a)
* 其中%s是读取方式，a是该文件
* 4.随机数的应用
* 首先，需要加入随机数种子
* eg：
* srand((unsigned)time(NULL));
* 这是采用的以系统时间所实现的随机数
* 其次，如果想要获得一定范围内的随机数
* 采用的是 [a,b)
* [rand（）%(b-a)]+a
* 通过数学的方法进行理解就是取模，然后再加
* 5.system，在 头文件 cstdlib中存储
* 调用方式为
* System("命令 （有空格） 加上文件")
* 6.文件打开
* if((rt=fopen("abc.txt","r"))==NULL)

# 更新内容

* While()的使用可以在一定程度上解决for循环的定量分析
* While 可以不用知道数量的多少
* 与此同时，我还学会了使用cmd的命令符

Dos 命令

dir (directory) ：列出当前目录下的文件以及文件夹

md (make directory)： 创建目录

rd (remove directory)：删除目录

cd (change directory)：进入指定目录

cd.. : 退回到上一级目录

cd\ : 退回到根目录

del：删除文件

exit：退出dos命令行

cls (clear screen): 清屏

* 一些其他的感想：
  1. 首先，在学习的过程中，必须要学会自学，而不是遇到不会的问题就要问别人，这是一个非常不好的习惯，尤其是对于一些大佬而言，他们非常厌烦将时间给那些不值得思考的问题，这简直就是浪费时间。所以我整理一些网站，非常有用：
     1. Cppreference
     2. Csdn
     3. 博客园
     4. Google
     5. 国内可以使用百度或者bing

另一方面，我们亦可以采用其他的方法进行纠错

其中，最重要的就是debug！！！！！！

对于dev-cpp这种不需要手动搭建环境的编译器，直接调用debug简直就是轻而易举

* 1. 其次，我们必须要通过项目来训练自己，而不是一味的跟随老师去做！！

比如，在自己编写排座位的小程序时，就会发现：

* + - 1. 用户手册是一个非常重要的东西，我们一定要从用户的角度考量自己的文件，而不是从一个程序员的角度进行考量
      2. 其次要注意尽量使用一种语言，尤其是不要跨语言！

比如使用py和cpp结合，虽然各有所长，但是实在时很难更正自己的语法

总之，学习c语言也好，还是其他语言也罢，自学才是学习计算机的真谛，没有老师会细致入微的讲解所有的语言，很难但更没必要，如果一个大学生还在纠结于老师讲的程度，那岂不是太菜了，不适合在这个知识爆炸的时代存活！！！

最后，我的感想就是：听十节课不如写一个程序，做十道题不如写一个项目！大学，必须学会面向结果编程！