

0x01 C语言——导论部分

为什么推荐先学C



VIDAR TEAM

C的优点

贴近硬件

功能强大

内容简洁

学习轻松

应用广泛

关注底层

为所欲为的快



C与安全

二进制安全



软件安全

IoT安全

系统安全



贴近底层

Web安全

打下基础



C语言

深入理解

高效学习



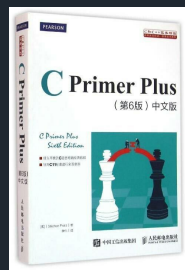
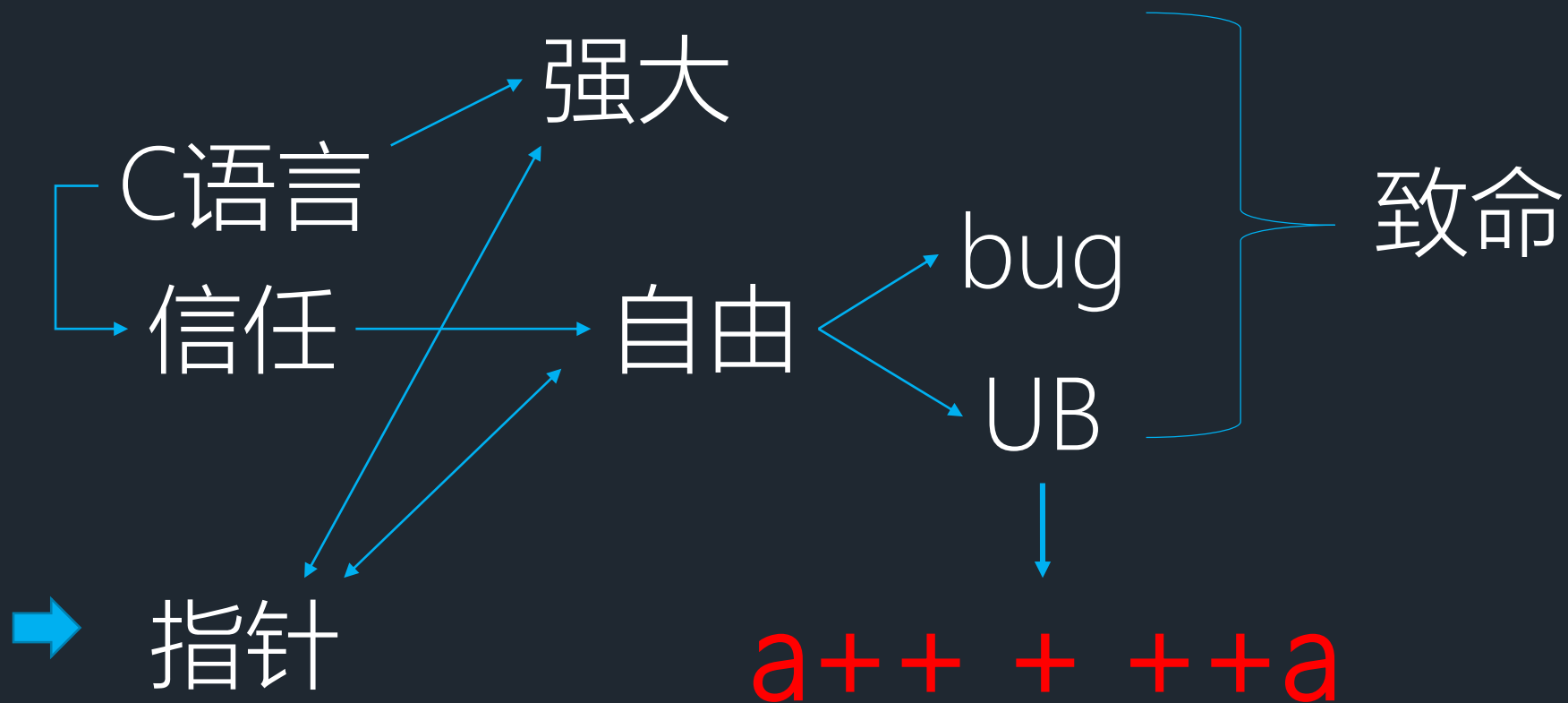
PHP Python Golang.....



攻防技术



详实



大多数人的信息课上应该提到过.....

机器语言→汇编语言→高级语言



好像除了名字以外啥都不知道子

Emm...起码我知道C是



VIDAR TEAM



贴近底层

Windows与Linux内核
都基于C语言开发

The Linux Kernel Archives

[About](#)[Contact us](#)[FAQ](#)[Releases](#)[Signatures](#)[Site news](#)

Protocol

Location

HTTP

<https://www.kernel.org/pub/>

GIT

<https://git.kernel.org/>

RSYNC

<rsync://rsync.kernel.org/pub/>

Latest Stable Kernel:

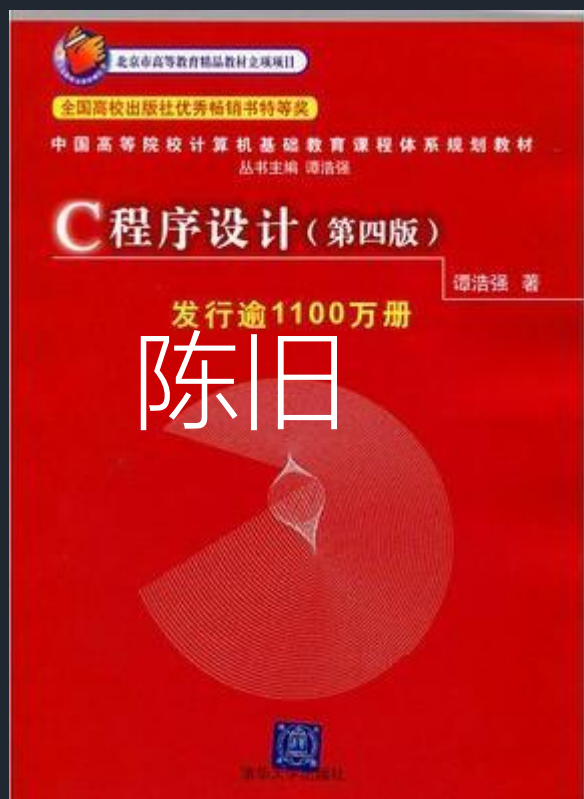
5.3.5

mainline:	5.4-rc2	2019-10-06	[tarball]	[patch]	[inc. patch]	[view diff]	[browse]
stable:	5.3.5	2019-10-07	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff] [browse] [changelog]
stable:	5.2.20 [EOL]	2019-10-07	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff] [browse] [changelog]
longterm:	4.19.78	2019-10-07	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff] [browse] [changelog]
longterm:	4.14.148	2019-10-07	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff] [browse] [changelog]
longterm:	4.9.196	2019-10-07	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff] [browse] [changelog]
longterm:	4.4.196	2019-10-07	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff] [browse] [changelog]
longterm:	3.16.75	2019-10-05	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff] [browse] [changelog]
linux-next:	next-20191011	2019-10-11					[browse]

acct.c	2019/7/8 6:41	C 源文件	16 KB
async.c	2019/7/8 6:41	C 源文件	11 KB
audit.c	2019/7/8 6:41	C 源文件	61 KB
audit.h	2019/7/8 6:41	H 文件	11 KB
audit_fsnotify.c	2019/7/8 6:41	C 源文件	6 KB
audit_tree.c	2019/7/8 6:41	C 源文件	26 KB
audit_watch.c	2019/7/8 6:41	C 源文件	15 KB
auditfilter.c	2019/7/8 6:41	C 源文件	34 KB
auditsc.c	2019/7/8 6:41	C 源文件	70 KB
backtracetest.c	2019/7/8 6:41	C 源文件	2 KB
bounds.c	2019/7/8 6:41	C 源文件	1 KB
capability.c	2019/7/8 6:41	C 源文件	15 KB
compat.c	2019/7/8 6:41	C 源文件	10 KB
configs.c	2019/7/8 6:41	C 源文件	3 KB
context_tracking.c	2019/7/8 6:41	C 源文件	7 KB
cpu.c	2019/7/8 6:41	C 源文件	57 KB
cpu_pm.c	2019/7/8 6:41	C 源文件	6 KB
crash_core.c	2019/7/8 6:41	C 源文件	12 KB
crash_dump.c	2019/7/8 6:41	C 源文件	2 KB
cred.c	2019/7/8 6:41	C 源文件	23 KB
delayacct.c	2019/7/8 6:41	C 源文件	5 KB
dma.c	2019/7/8 6:41	C 源文件	4 KB
elfcore.c	2019/7/8 6:41	C 源文件	1 KB
exec_domain.c	2019/7/8 6:41	C 源文件	2 KB
exit.c	2019/7/8 6:41	C 源文件	45 KB
extable.c	2019/7/8 6:41	C 源文件	5 KB
fail_function.c	2019/7/8 6:41	C 源文件	8 KB

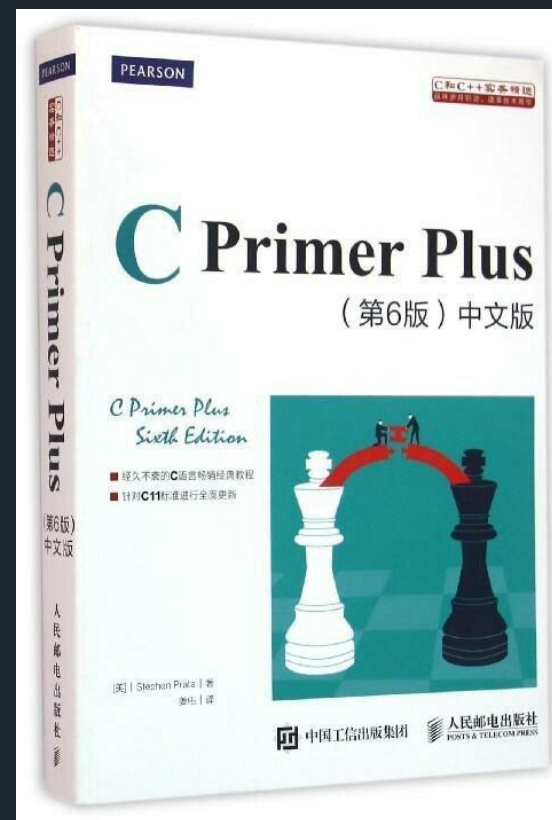


如何学习C语言



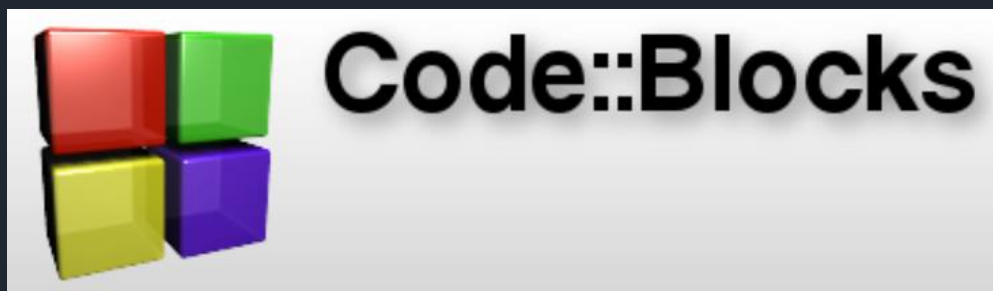
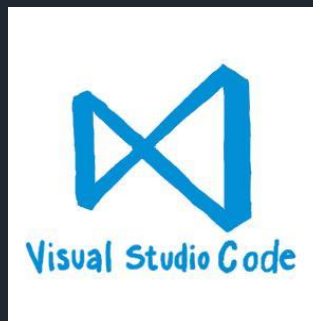
陈旧

浅薄
学校的书



VIDAR TEAM

用什么写C



为什么不推荐vc++ 6.0

Microsoft Visual C++ 6.0

编辑

收藏 279 52

Microsoft Visual C++，（简称Visual C++、MSVC、VC++或VC）是Microsoft公司推出的以C++语言为基础的开发Windows环境程序，面向对象的可视化集成编程系统。它不但具有程序框架自动生成、灵活方便的类管理、代码编写和界面设计集成交互操作、可开发多种程序等优点，而且通过的设置就可使其生成的程序框架支持数据库接口、OLE2.0，WinSock网络。^[1]

Microsoft Visual C++ 6.0，简称VC6.0，是微软于1998年推出的一款C++编译器，集成了MFC 6.0，包含标准版（Standard Edition）、专业版（Professional Edition）与企业版（Enterprise Edition）^[2]。发行至今一直被广泛地用于大大小小的项目开发。（但是，这个版本在Windows XP下运行会出现问题，尤其是在调试模式的情况下（例如：静态变量的值并不会显示）。这个调试问题可以通过打一个叫“Visual C++ 6.0 Processor Pack”的补丁^[3]来解决。）

Microsoft Visual C++ 6.0对windows7和windows8的兼容性较差。在Windows7使用VC6.0只需要忽略兼容性提示即可正常使用，^[4]但是在Windows8(含Windows8.1)使用VC6.0则需要改原MSDEV.EXE文件名并改兼容性才能正常使用。^[5]在Windows10的第一个版本也可以正常使用VC6.0，但Windows10系统升级更新后中文版VC6.0无法正常使用，提示“0xc0000142”的错误，需要将原MSDEV.EXE文件替换为英文版或者汉化版才能正常使用^[6]（该版本在Windows8(含Windows8.1)照样需要改原MSDEV.EXE文件名并改兼容性才能正常使用。）。

软件名称	Microsoft Visual C++ 6.0	更新时间	1998年
开发商	微软	软件语言	英语
软件平台	Windows 95/98SE/2000/NT 4.0	软件类型	集成开发环境（IDE）
软件版本	6.0	系列最新版	Microsoft Visual C++ 2017



farta

908 人赞同了该回答

这是好事啊，好得很。VC6还不够，最好用TC，关门放谭浩强，int得是16位的，main函数得是void的，wchar_t是不存在的，什么C++标准，什么namespace，都打翻在地批倒批臭，什么i++++++i，能教多细就教多细，绕死那帮学生，让他们毛都学不会，毕业出来面试时只能对着编译错误抓狂，这样我们这些老鸟才能名正言顺要求高工资。

至于那些会自己安装VS 2017和gcc，自己会学c++14的人，他们就算不是天才，至少也基本注定了是能在业内混下去的人，根本不用任何人担心。

发布于 2017-03-31

已赞同 908

117 条评论

分享

收藏

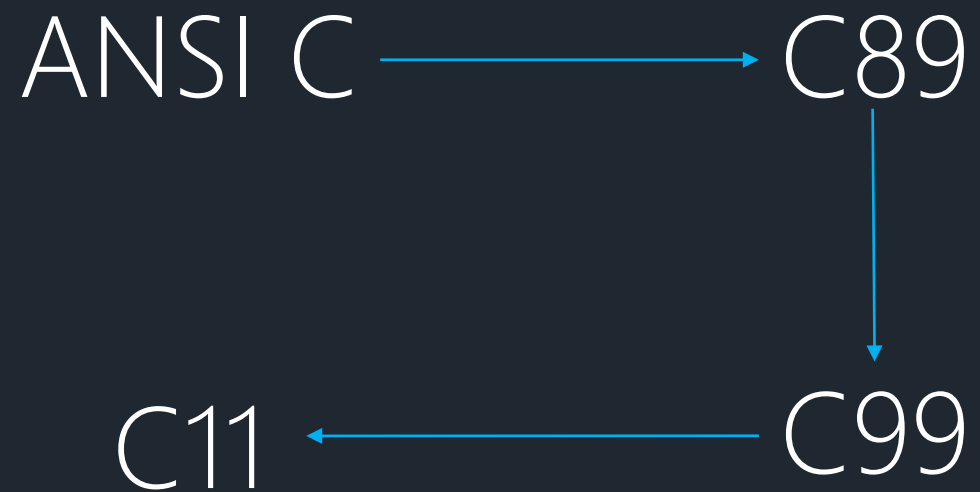
感谢



...



VIDAR TEAM

紧跟潮流



 百度一下

Google 搜索

手气不错

Google 提供: English

segmentfault

首页

问答

专栏

讲堂

发现

在 SegmentFault, 学习技能、解决问题

每个月, 我们帮助 1000 万的开发者解决各种各样的技术问题。并助力他们在技术能力、职业生涯、影响力上获得提升。

为你推荐

为你推荐

stackoverflow

Search...

Learn, Share, Build

Each month, over 50 million developers come to Stack Overflow to learn, share their knowledge, and build their careers.

Join the world's largest developer community.



VIDAR TEAM

函数导论



VIDAR TEAM

什么是函数

· 是一个大型程序中的某部分代码，由一个或多个语句块组成。它负责完成某项有意义的任务——通常是处理文本，控制输入或计算数值。

· 函数相较于其他代码，具备相对的独立性。



VIDAR TEAM

函数的作用

在多个程序之间重复使用代码

减少程序中的重复代码

把程序划分成不同作用的功能模块

让程序变得简洁，提高可读性



VIDAR TEAM

```
for (size_t i = 0; i < infix_input.length(); i++) {  
    if (is_num(infix_input[i])) {//为数字  
        out.push(infix_input[i]);  
    }  
    else if (is_op(infix_input[i])) {//为操作符  
        out.push(' ');  
        if (op.is_empty() || op.get_top_elem() == '(')  
            op.push(infix_input[i]);  
    }  
}
```



0x02 C语言——深入部分

还记得国庆解谜的小游戏吗



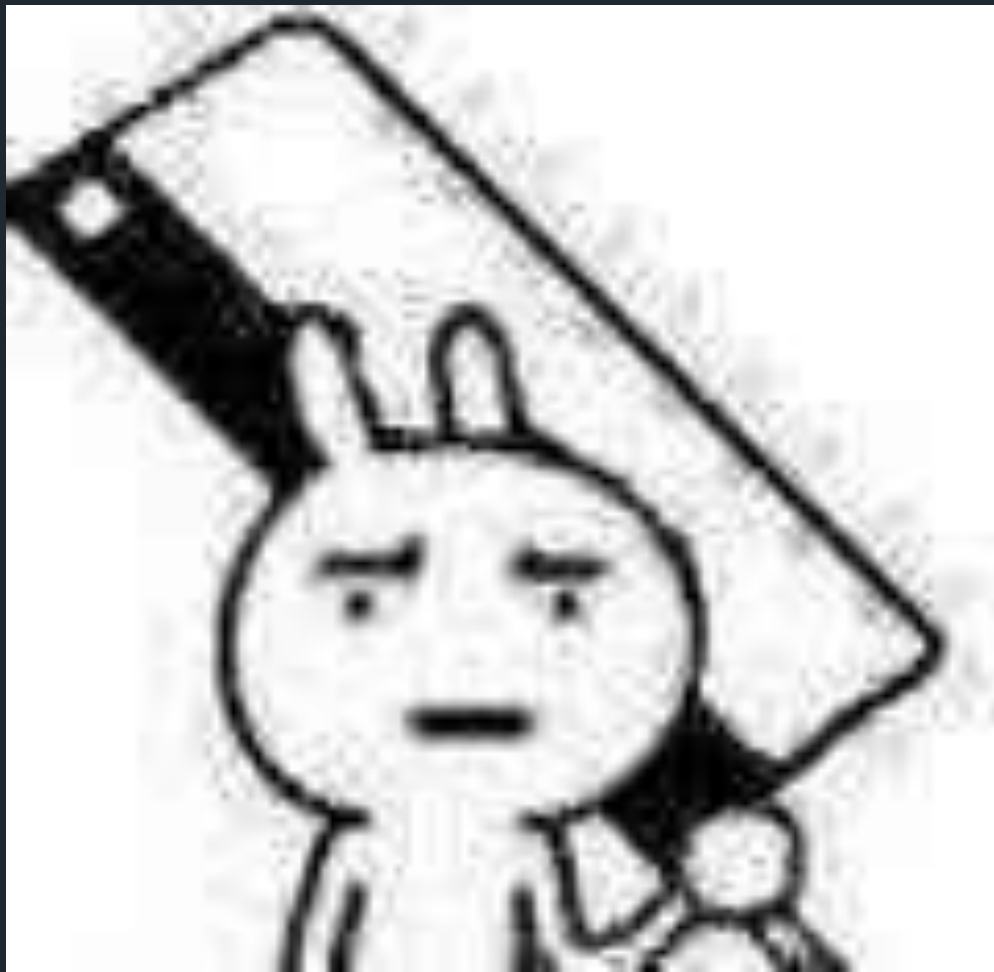
选择D:\Download\c (1).exe

欢迎来到Vidar Shop

在这里，你要面对Li4n0的压榨，每个月房租的紧逼，生命值逐渐的下降
同时也能了解很多的学长（的口头禅）
靠着努力或者智慧，你一定能得到flag

E1 Psy Congroo!

（这是一个不欢迎浮点数的游戏，若输入浮点数后果自负2333
请按任意键继续. . .



首先我要对用CE等直接修改内存做题的同学表达赞赏

今日的Vidar Shop开始营业了,店长Li4n0对你实行了惨绝人寰的9127工作制。

现在你的生命值为: 200

今日进货的商品如下:

1. 传说中的Beats Solo3 听说市场价1688
2. 正常的稳赚学长的西瓜战车
3. 战痕累累的华为Mate30
4. 健康要紧, 劳资不干了!!!

你想要卖哪个？

1

这时来了一位富有的大爷
说出你想开的价格

1

出售成功！

当前财产：100001

你想要卖哪个？

2

这时来了一位贫困的Li4n0

说出你想开的价格

-100001

出售成功！

当前财产： 0

你想要卖哪个？

3

这时来了一位贫困的大妈

说出你想开的价格

-1

出售成功！

当前财产：18446744073709551615

今日的工作结束了，由于Li4n0的虐待，你的生命值下降了10点

当前生命值：190

扣除一日的房租：3000

当前资产为：18446744073709548615


请按任意键继续. . . ■



VIDAR TEAM

unsigned / signed

```
int main()
{
    int a = -1;
    printf("%d\n%u", a, a);
    return 0;
}
```

 Microsoft Vis

-1
4294967295

"%d\n%u"

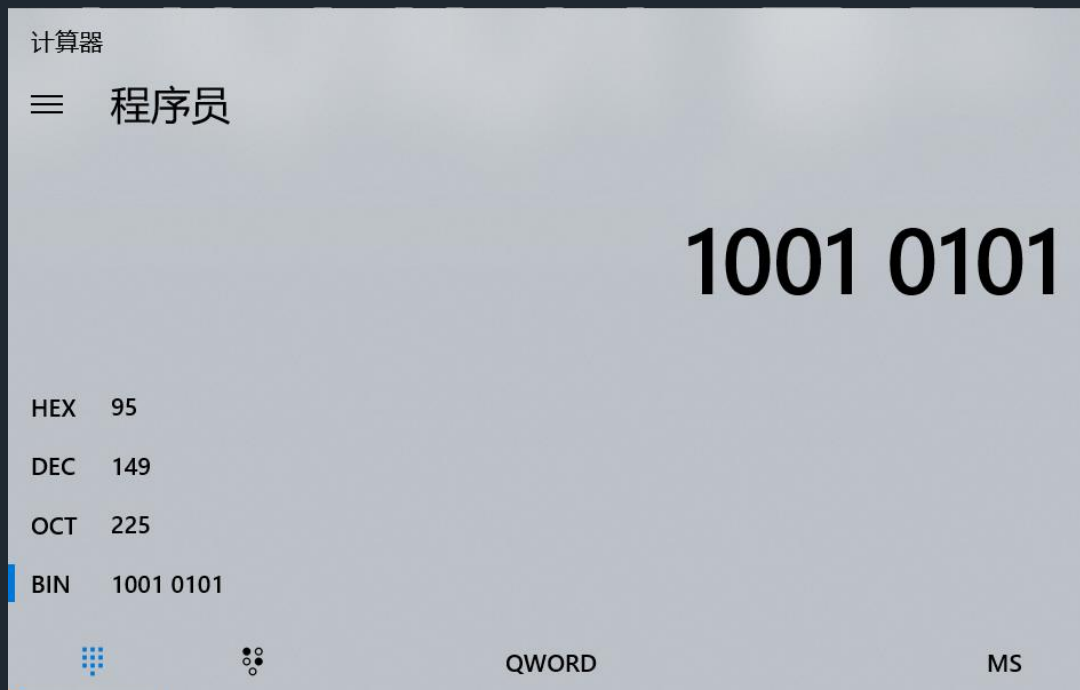
从C语言到二进制

理解表象背后的本质

(老少皆宜的超超超入门版)



10010101 B = ??? D
10010101 B = 149 D



HEX	95
DEC	149
OCT	225
BIN	1001 0101

二进制中如何表示负数？

计算器

≡ 程序员

-1001 0101

HEX 95

DEC 149

OCT 225

BIN 1001 0101



VIDAR TEAM

1 0010101



符号位

具体实现可以搜索“补码”

%u: 计算机认为这是个无符号数 149

%d: 计算机认为这是个有符号数 -106

C语言允许直接转换类型

```
int a=97;      float a=1.1;  
printf("%c",a); printf("%d",a);
```

数据类型的本质

存储大小

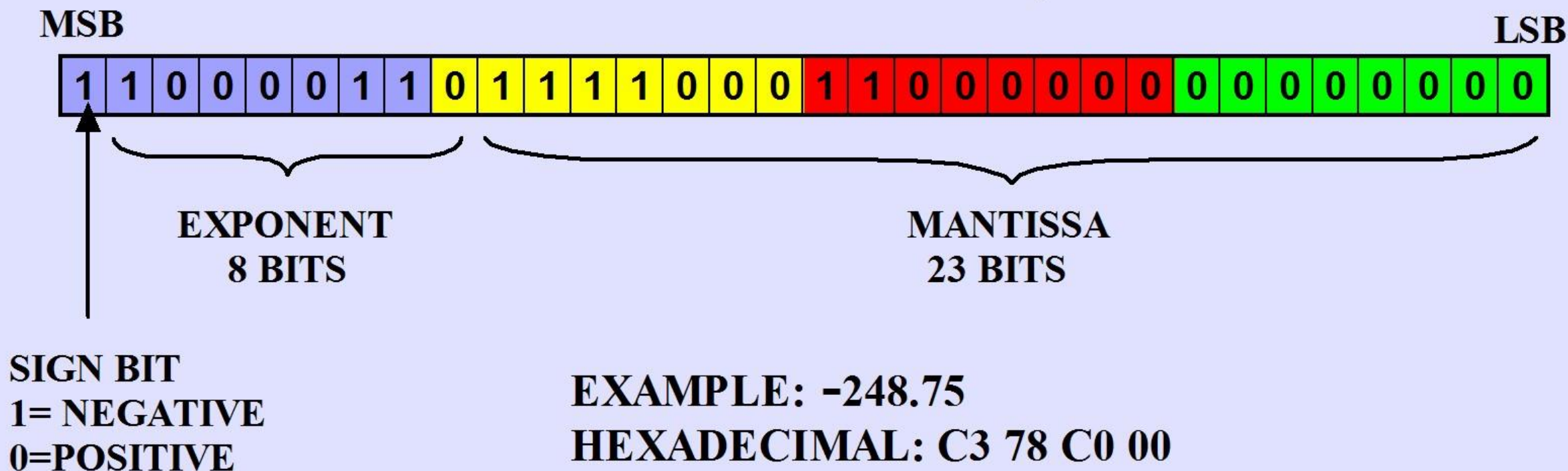
存储格式

int	4 Byte
char	1 Byte
Float	4 Byte
Double	8 Byte

signed int
char
float

补码
ASCII
?

FLOATING POINT FORMAT IEEE-754, 32 BITS



有些复杂，若想一探浮点数究竟
欢迎课后自行学习或者群内提问

```
int a = 1;
```

变量的实质

类型

int

值

1

地址

&a

unsigned / signed

```
long long price;
```

```
puts("说出你想开的价格");  
scanf_s("%lld", &price);
```

```
unsigned long long uprice = (unsigned long long)price;  
yourmoney += uprice;
```



VIDAR TEAM

同样的数据，却能有多种输出的结果
计算机只认0和1，而理解的方式由人规定

计算机中本只有0和1
理解的方式多了
便成了数据类型

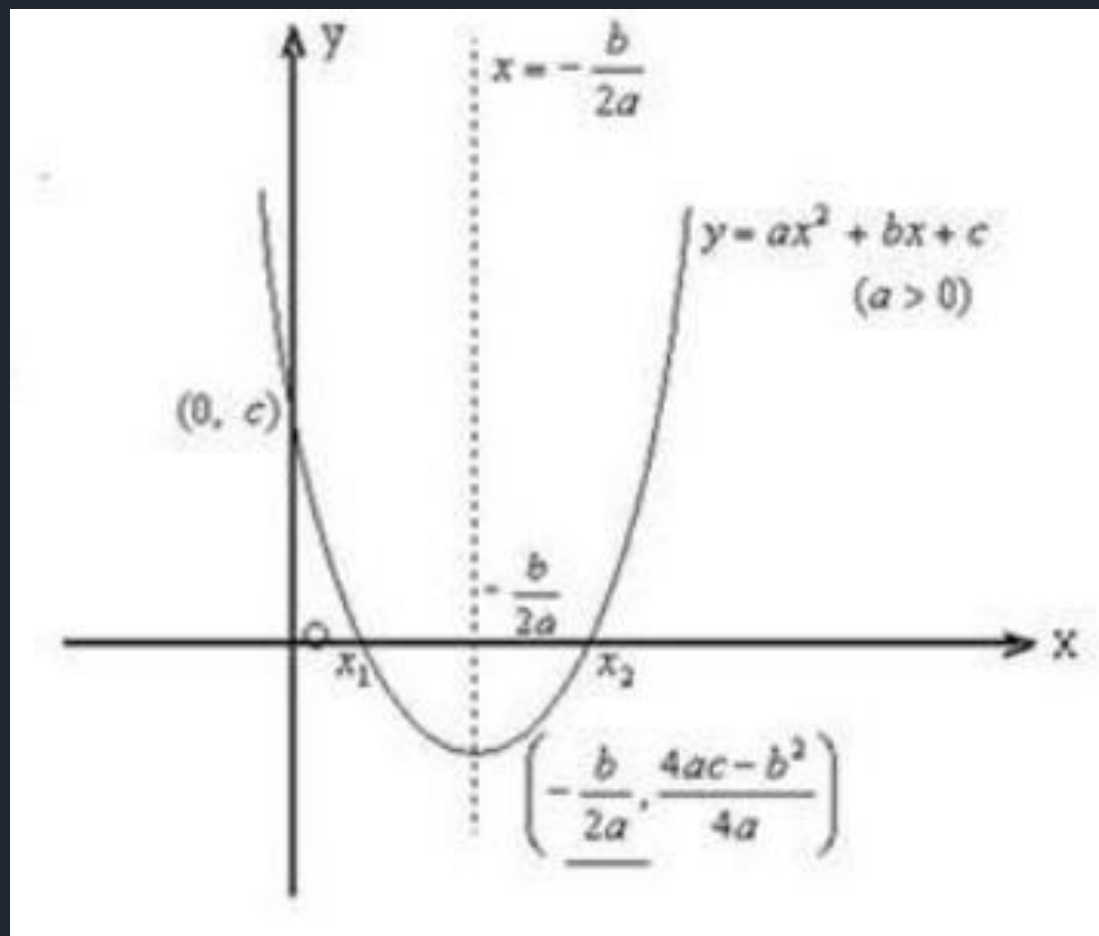


指针：从入门到深入精通

(老少皆宜的超超超入门版)



指针简介-高度抽象化



这是一个指针：

$$\text{顶点坐标: } \left(-\frac{b}{2a}, \frac{4ac - b^2}{4a}\right)$$

该指针指向的值：

$$y_{\text{值}} = \frac{4ac - b^2}{4a}$$

首先你得知道.....

&: 取地址运算符, 获取变量的实际地址

```
int num = 1;  
int* a = &num;  
int* b = 0;
```

本质上都是在对指针变量进行赋值,
只是&num是有效地址, 0是无效地址

首先你得知道.....

★：解引用运算符，将操作数当做地址来获得其指向的对象

```
int a = 1;
```

```
int* ptr = &a;
```

```
int b = *ptr;
```

~~ 倘若解引用的对象不是指针变量.....

```
int a = 0x12345, b=0;
```

```
int c = *a;
```

```
int d = *b;
```

Just like.....

Minecraft 创造模式

指针：自由，强大，有趣

开局先来一个基础四连

```
int a = 114514;  
int * ptr = &a;  
printf("%d\n", a);  
printf("%d\n", ptr);  
printf("%p\n", ptr);  
printf("%d\n", *ptr);
```

再抽取一位
幸运观众



你可能觉得自己好像懂了.....

```
int * a = 0;  
printf("%d",a+1);
```

没事，此时不懂很正常，指针的学习并非一朝一夕之功，而成功与否的关键在于你的思考与实践

思考.....

```
int a=0;
```

```
scanf("%d" , &a);
```

为什么是&a而不是a?



收快递

```
printf("%d" , a);
```

这里为什么又不需要&a?



发快递

核心概念

指向的类型

存储的地址

指针变量与指针

存储的地址

- 内存中任意位置都拥有独一无二的坐标，一般称其为地址
- 同一地址不可能同时存放两个数据

不然就穿模了

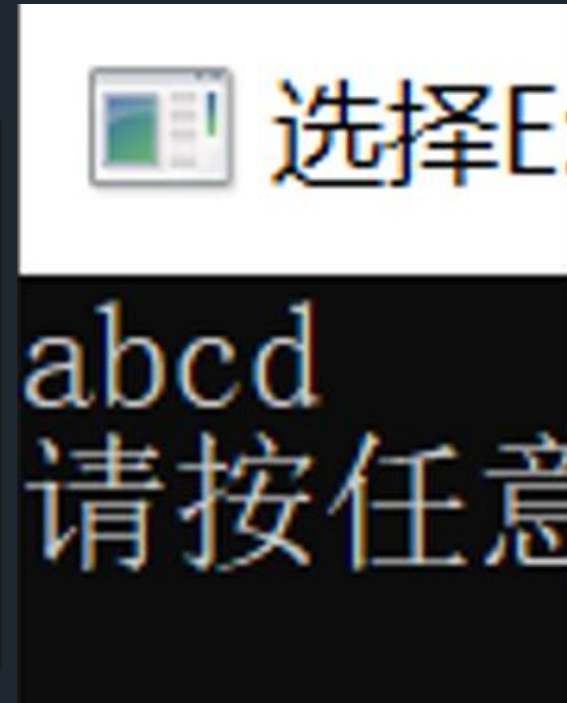


指向的类型

如果类型使用错误会怎样？

尝试一下：

```
char* str = "abcd";  
int* a = str;  
printf("%s\n", a);
```



指向int的指针变量被赋值指向char类型的指针时，仍然能用于输出该字符串

指向的类型

好像没怎么样？

指向的类型

指针本质不就是个地址，为什么还要知道指向的对象类型？

只要地址正确，无论给的指针是指向何种类型应该都无所谓？

回到之前.....

```
int * a = 0;  
printf("%d",a+1);
```

对指针直接加减 1 的意义？

**寻找该地址前/后一项
同类型数据的位置**

再回到之前.....

常见数据类型存储大小

int	4 Byte
char	1 Byte
Float	4 Byte
Double	8 Byte

总结：指向的类型

- 指针寻找下一项的地址时，需要知道地址是加 1 还是 4 还是 8 还是.....
- 方便自身记忆

指针变量与指针

```
int a = 0;  
int* ptr = &a;  
int** ppttr = &ptr;  
printf("0x%p\n", ptr);
```

0x00AFFAF4

请按任意键继续

名词归纳环节.....

指针

指针变量

地址

指向指针变量的指针

ptr

指针变量

ppttrr

(双重指针) 指向指针变量的指针

0x00AFFAF4

地址

指针变量存储的值是指针
其本质也是变量，因此存在地址，
即&ptr

地址与指针

0xAFFAF4 是变量a的地址
是指向a的指针

抽象思想

指针是对内存地址概念的抽象化



语言层面
(浅)



计算机底层
(深)



计算机科学中的
重要思想