Lecture 1+

Preliminary

Something about the C

How much you know C

- A simple test

31 - How many parameters in a function definition is supported at most?

31 - How many arguments in a function call is supported at most?

509 - How many characters in a source line is supported at most?

32 - How many levels of nested parentheses in an expression is supported at most?

- What is the maximum value for a **long int**? (hint: 32 bits) 2147483647 0x7fffffff

How much you know C

- A simple test

typedef struct bar {int bar;} bar

How to transfer a multi-dimension array with different rows to a function?

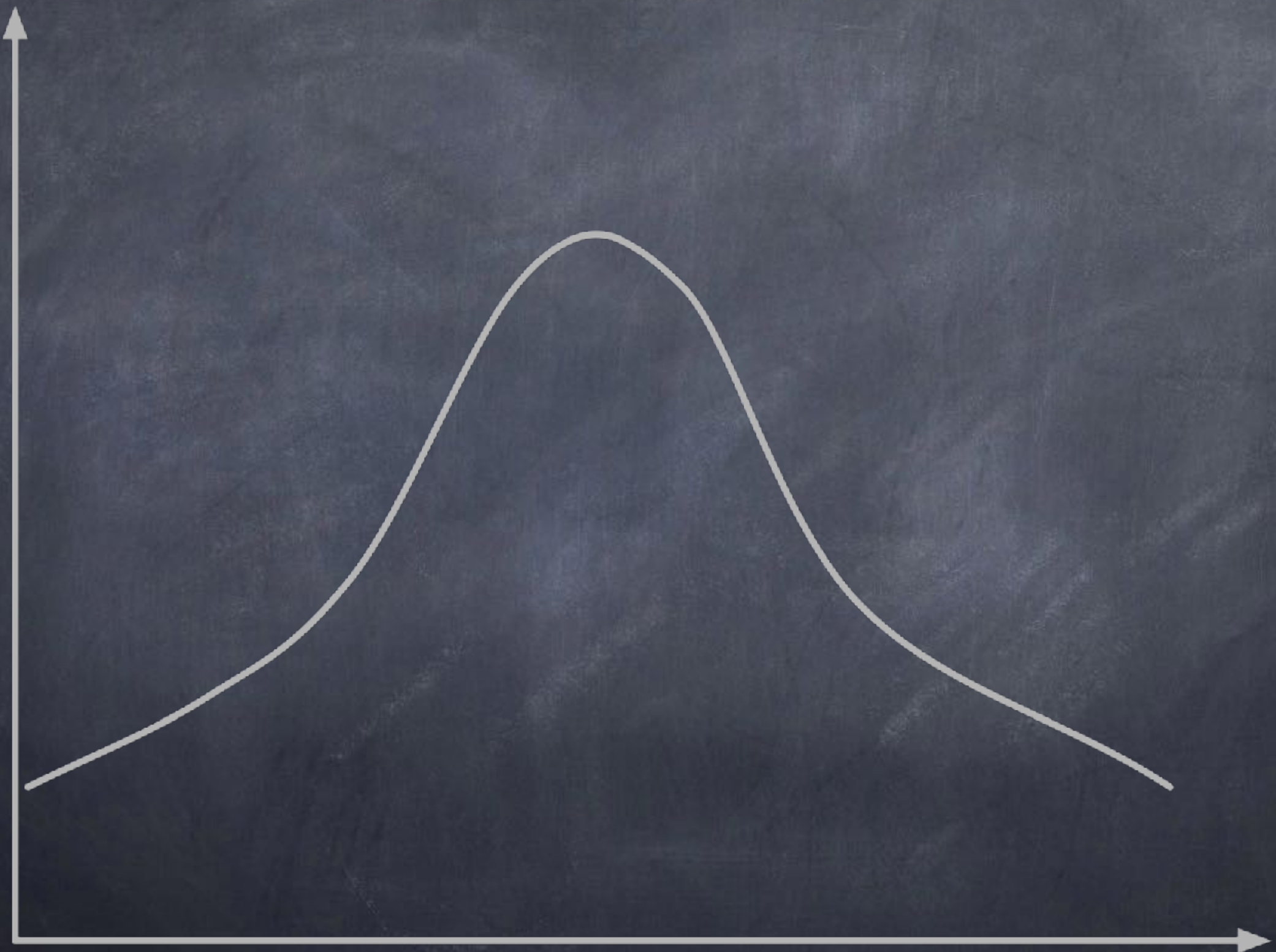
extern char *p doesn't match char p[100] in another file?

What's the difference between
char *foo[] & char (*foo)[] ?

How much you know C

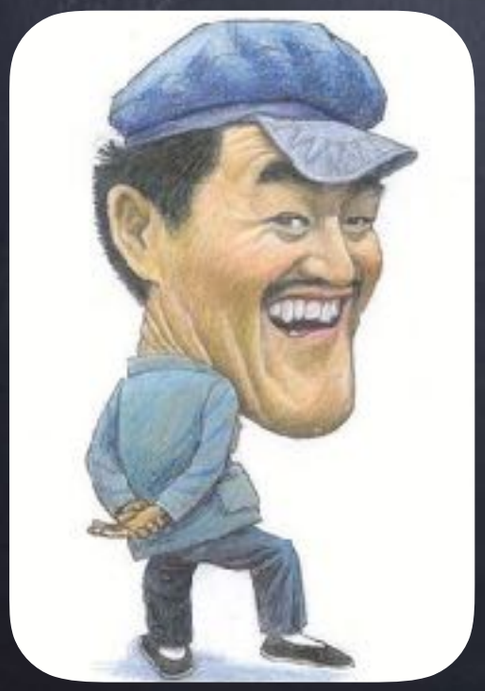
- An investigation

What's the most annoyed things to YOU referring to C?



How much you know C

```
#include                                    <math.h>
#include                                   <sys/time.h>
#include                                   <X11/Xlib.h>
#include                                  <X11/keysym.h>
                                          double L ,o ,P
                                         ,_=dt,T,Z,D=1,d,
                                         s[999],E,h= 8,I,
                                         J,K,w[999],M,m,O
                                        ,n[999],j=33e-3,i=
                                        1E3,r,t, u,v ,W,S=
                                        74.5,l=221,X=7.26,
                                        a,B,A=32.2,c, F,H;
                                        int N,q, C, y,p,U;
                                      Window z; char f[52]
                                    ; GC k; main(){ Display*e=
 XOpenDisplay( 0); z=RootWindow(e,0); for (XSetForeground(e,k=XCreateGC (e,z,0,0),BlackPixel(e,0))
; scanf("%lf%lf%lf",y +n,w+y, y+s)+1; y ++); XSelectInput(e,z= XCreateSimpleWindow(e,z,0,0,400,400,
0,0,WhitePixel(e,0) ),KeyPressMask); for(XMapWindow(e,z); ; T=sin(O)){ struct timeval G={ 0,dt*1e6}
; K= cos(j); N=1e4; M+= H*_; Z=D*K; F+=_*P; r=E*K; W=cos( O); m=K*W; H=K*T; O+=D*_*F/ K+d/K*E*_; B=
sin(j); a=B*T*D-E*W; XClearWindow(e,z); t=T*E+ D*B*W; j+=d*_*D-_*F*E; P=W*E*B-T*D; for (o+=(I=D*W+E
*T*B,E*d/K *B+v+B/K*F*D)*_; p<y; ){ T=p[s]+i; E=c-p[w]; D=n[p]-L; K=D*m-B*T-H*E; if(p [n]+w[ p]+p[s
]== 0|K <fabs(W=T*r-I*E +D*P) |fabs(D=t *D+Z *T-a *E)> K)N=1e4; else{ q=W/K *4E2+2e2; C= 2E2+4e2/ K
 *D; N-1E4&& XDrawLine(e ,z,k,N ,U,q,C); N=q; U=C; } ++p; } L+=_* (X*t +P*M+m*l); T=X*X+ l*l+M *M;
  XDrawString(e,z,k ,20,380,f,17); D=v/l*15; i+=(B *l-M*r -X*Z)*_; for(; XPending(e); u *=CS!=N){
                                    XEvent z; XNextEvent(e ,&z);
                                        ++*((N=XLookupKeysym
                                          (&z.xkey,0))-IT?
                                          N-LT? UP-N?& E:&
                                          J:& u: &h); --*(
                                          DN -N? N-DT ?N==
                                          RT?&u: & W:&h:&J
                                           ); } m=15*F/l;
                                           c+=(I=M/ l,l*H
                                           +I*M+a*X)*_; H
                                           =A*r+v*X-F*l+(
                                           E=.1+X*4.9/l,t
                                           =T*m/32-I*T/24
                                            )/S; K=F*M+(
                                            h* 1e4/l-(T+
                                            E*5*T*E)/3e2
                                            )/S-X*d-B*A;
                                            a=2.63 /l*d;
                                            X+=( d*l-T/S
                                             *(.19*E +a
                                             *.64+J/1e3
                                             )-M* v +A*
                                             Z)*_; l +=
                                             K *_; W=d;
                                             sprintf(f,
                                             "%5d  %3d"
                                             "%7d",p =l
                                            /1.7,(C=9E3+
 O*57.3)%0550,(int)i); d+=T*(.45-14/l*
X-a*130-J* .14)*_/125e2+F*_*v; P=(T*(47
*I-m* 52+E*94 *D-t*.38+u*.21*E) /1e2+W*
179*v)/2312; select(p=0,0,0,0,&G); v-=(
 W*F-T*(.63*m-I*.086+m*E*19-D*25-.11*u
  )/107e2)*_; D=cos(o); E=sin(o); } }
```



How much you know C

```
/*!
 * Bootstrap Responsive v2.0.4
 *
 * Copyright 2012 Twitter, Inc
 * Licensed under the Apache Lice
 * http://www.apache.org/licenses,
 *
 * Designed and built with all th
 */

.clearfix {
  *zoom: 1;
}

.clearfix:before,
.clearfix:after {
  display: table;
  content: "";
}

.clearfix:after {
  clear: both;
}

.hide-text {
  font: 0/0 a;
  color: transparent;
  text-shadow: none;
  background-color: transparent;
  border: 0;
}

.input-block-level {
  display: block;
  width: 100%;
  min-height: 28px;
  -webkit-box-sizing: border-box;
     -moz-box-sizing: border-box;
      -ms-box-sizing: border-box;
          box-sizing: border-box;
}

.hidden {
  display: none;
  visibility: hidden;
}

.visible-phone {
  display: none !important;
}
```

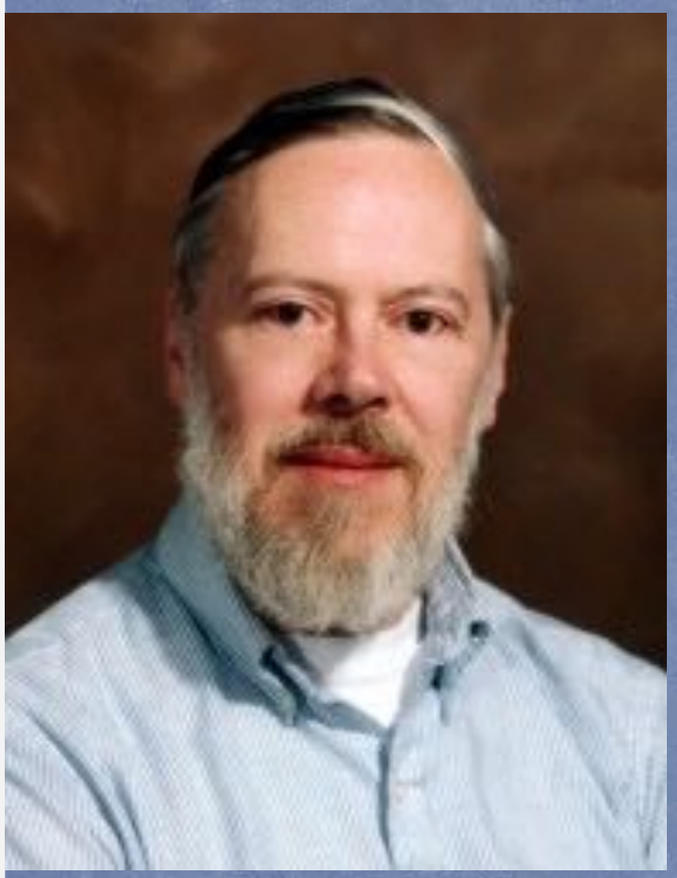
```
/*!
 * Bootstrap Responsive v2.0.4
 *
 * Copyright 2012 Twitter, Inc
 * Licensed under the Apache License v2.0
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Designed and built with all the love in the world @twitter by @mdo and @fat.
 */.clearfix{*zoom:1}.clearfix:before,.clearfix:after{display:table;content:""}.clearfix:after{clear:both}.hide-
text{font:0/0 a;color:transparent;text-shadow:none;background-color:transparent;border:0}.input-block-
level{display:block;width:100%;min-height:28px;-webkit-box-sizing:border-box;-moz-box-sizing:border-box;-ms-box-
sizing:border-box;box-sizing:border-box}.hidden{display:none;visibility:hidden}.visible-phone{display:none!
important}.visible-tablet{display:none!important}.hidden-desktop{display:none!important}@media(max-width:767px)
{.visible-phone{display:inherit!important}.hidden-phone{display:none!important}.hidden-desktop{display:inherit!
important}.visible-desktop{display:none!important}}@media(min-width:768px) and (max-width:979px){.visible-
tablet{display:inherit!important}.hidden-tablet{display:none!important}.hidden-desktop{display:inherit!
important}.visible-desktop{display:none!important}}@media(max-width:480px){.nav-collapse{-webkit-
transform:translate3d(0,0,0)}.page-header h1 small{display:block;line-height:18px}
input[type="checkbox"],input[type="radio"]{border:1px solid #ccc}.form-horizontal .control-
group>label{float:none;width:auto;padding-top:0;text-align:left}.form-horizontal .controls{margin-left:0}.form-
horizontal .control-list{padding-top:0}.form-horizontal .form-actions{padding-right:10px;padding-left:
10px}.modal{position:absolute;top:10px;right:10px;left:10px;width:auto;margin:0}.modal.fade.in{top:auto}.modal-
header .close{padding:10px;margin:-10px}.carousel-caption{position:static}}@media(max-width:767px){body{padding-right:
20px;padding-left:20px}.navbar-fixed-top,.navbar-fixed-bottom{margin-right:-20px;margin-left:-20px}.container-
fluid{padding:0}.dl-horizontal dt{float:none;width:auto;clear:none;text-align:left}.dl-horizontal dd{margin-left:
0}.container{width:auto}.row-fluid{width:100%}.row,.thumbnails{margin-left:0}[class*="span"],.row-fluid [class*="span"]
{display:block;float:none;width:auto;margin-left:0}.input-large,.input-xlarge,.input-
xxlarge,input[class*="span"],select[class*="span"],textarea[class*="span"],.uneditable-input{display:block;width:
100%;min-height:28px;-webkit-box-sizing:border-box;-moz-box-sizing:border-box;-ms-box-sizing:border-box;box-
sizing:border-box}.input-prepend input,.input-append input,.input-prepend input[class*="span"],.input-append
input[class*="span"]{display:inline-block;width:auto}}@media(min-width:768px) and (max-width:979px){.row{margin-
left:-20px;*zoom:1}.row:before,.row:after{display:table;content:""}.row:after{clear:both}[class*="span"]
{float:left;margin-left:20px}.container,.navbar-fixed-top .container,.navbar-fixed-bottom .container{width:
724px}.span12{width:724px}.span11{width:662px}.span10{width:600px}.span9{width:538px}.span8{width:476px}.span7{width:
414px}.span6{width:352px}.span5{width:290px}.span4{width:228px}.span3{width:166px}.span2{width:104px}.span1{width:
42px}.offset12{margin-left:764px}.offset11{margin-left:702px}.offset10{margin-left:640px}.offset9{margin-left:
578px}.offset8{margin-left:516px}.offset7{margin-left:454px}.offset6{margin-left:392px}.offset5{margin-left:
330px}.offset4{margin-left:268px}.offset3{margin-left:206px}.offset2{margin-left:144px}.offset1{margin-left:82px}.row-
fluid{width:100%;*zoom:1}.row-fluid:before,.row-fluid:after{display:table;content:""}.row-fluid:after{clear:both}.row-
fluid [class*="span"]{display:block;float:left;width:100%;min-height:28px;margin-left:2.762430939%;*margin-left:
2.709239449638298%;-webkit-box-sizing:border-box;-moz-box-sizing:border-box;-ms-box-sizing:border-box;box-
sizing:border-box}.row-fluid [class*="span"]:first-child{margin-left:0}.row-fluid .span12{width:99.999999993%;*width:
99.9468085036383%}.row-fluid .span11{width:91.436464082%;*width:91.38327259263829%}.row-fluid .span10{width:
82.87292817100001%;*width:82.8197366816383%}.row-fluid .span9{width:74.30939226%;*width:74.25620077063829%}.row-
fluid .span8{width:65.74585634900001%;*width:65.6926648596383%}.row-fluid .span7{width:57.18232043800005%;*width:
57.129128948638304%}.row-fluid .span6{width:48.618784527%;*width:48.5655930376383%}.row-fluid .span5{width:
40.055248616%;*width:40.0020571266383%}.row-fluid .span4{width:31.491712705%;*width:31.4385212156383%}.row-
fluid .span3{width:22.928176794%;*width:22.874985304638297%}.row-fluid .span2{width:14.364640883%;*width:
14.311449393638298%}.row-fluid .span1{width:5.801104972%;*width:5.747913482638298%}input,textarea,.uneditable-
input{margin-left:0}input.span12,textarea.span12,.uneditable-input.span12{width:714px}
input.span11,textarea.span11,.uneditable-input.span11{width:652px}input.span10,textarea.span10,.uneditable-
```

Why C

When was C born?

C is quirky, flawed, and an enormous success

Note



C语言和Unix创始人丹尼斯·里奇（Dennis Ritchie）已于2011年10月9日逝世，享年70岁。

里奇于1965年获得哈弗大学物理学博士学位，1967年进入贝尔实验室（Bell Labs），开始了他杰出的职业生涯。1969年，他与肯·汤普逊（Ken Thompson）及其他贝尔实验室人员一起开发Unix操作系统。大约就在那时，他开始研发程序设计语言（即后来的C语言）。1978年，他与汤普逊出版了著名的《C程序设计语言》。

里奇获得的荣誉有：1983年荣获图灵奖，1988年当选美国国家工程院院士以及1999年荣获美国国家技术勋章等等。

When was C born?

The $20 Million Bug

In Spring 1993, in the Operating System development group at SunSoft, we had a "priority one" bug report come in describing a problem in the asynchronous I/O library. The bug was holding up the sale of $20 million worth of hardware to a customer who specifically needed the library functionality, so we were extremely motivated to find it. After some intensive debugging sessions, the problem was finally traced to a statement that read :

```
x==2;
```

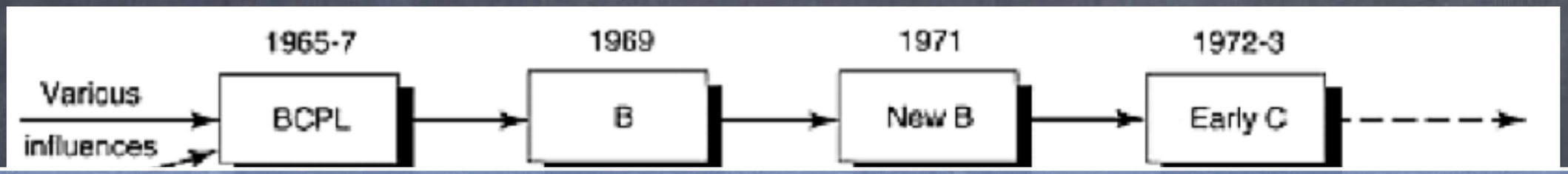
```
x == 2;
```

When was C born?

- The story of C begins with a failure.
- In 1969 the great Multics project—a joint venture between General Electric, MIT, and Bell Laboratories to build an operating system—was clearly in trouble.
- After that, Ken Thompson and Dennis Ritchie amused themselves porting Thompson's "Space Travel" software to a little-used PDP-7, deriving a new operating system, much simpler and lighter-weight than Multics, written in assembler language.

When was C born?

- Brian Kernighan coined the name "UNIX" in 1970.



Note

"BCPL: A Tool for Compiler Writing and System Programming," Martin Richards, Proc. AFIPS Spring Joint Computer Conference, 34 (1969), pp. 557-566. BCPL is not an acronym for the "Before C Programming Language", though the name is a happy coincidence. It is the "Basic Combined Programming Language"— "basic" in the sense of "no frills"—and it was developed by a combined effort of researchers at London University and Cambridge University in England. A BCPL implementation was available on Multics.

When was C born?

- In this potential chicken-and-egg situation, UNIX definitely came well before C (and it's also why UNIX system time is measured in seconds since January 1, 1970—that's when time began).
- B retained the typelessness of BCPL; the only operand was a machine word.
- Dennis Ritchie capitalized on the more powerful PDP-11 to create "New B," which solved both problems, multiple datatypes, and performance.

Software Dogma 1

The Golden Rule of Compiler-Writers: Performance Is (almost) Everything.

- Performance is almost everything in a compiler. Compiler performance has two aspects: runtime performance (how fast the code runs) and compile time performance (how long it takes to generate code). Runtime performance usually dominates, except in development and student environments.

Software Dogma 1

The Golden Rule of Compiler-Writers: Performance Is (almost) Everything.

- Many compiler optimizations cause longer compilation times but make run times much shorter. Other optimizations (such as dead code elimination, or omitting runtime checks) speed up both compile time and run time, as well as reducing memory use. The downside of aggressive optimization is the risk that invalid results may not be flagged. Optimizers are very careful only to do safe transformations, but programmers can trigger bad results by writing invalid code (e.g., referencing outside an array's bounds because they "know" that the desired variable is adjacent).

When was C born?

- the early C

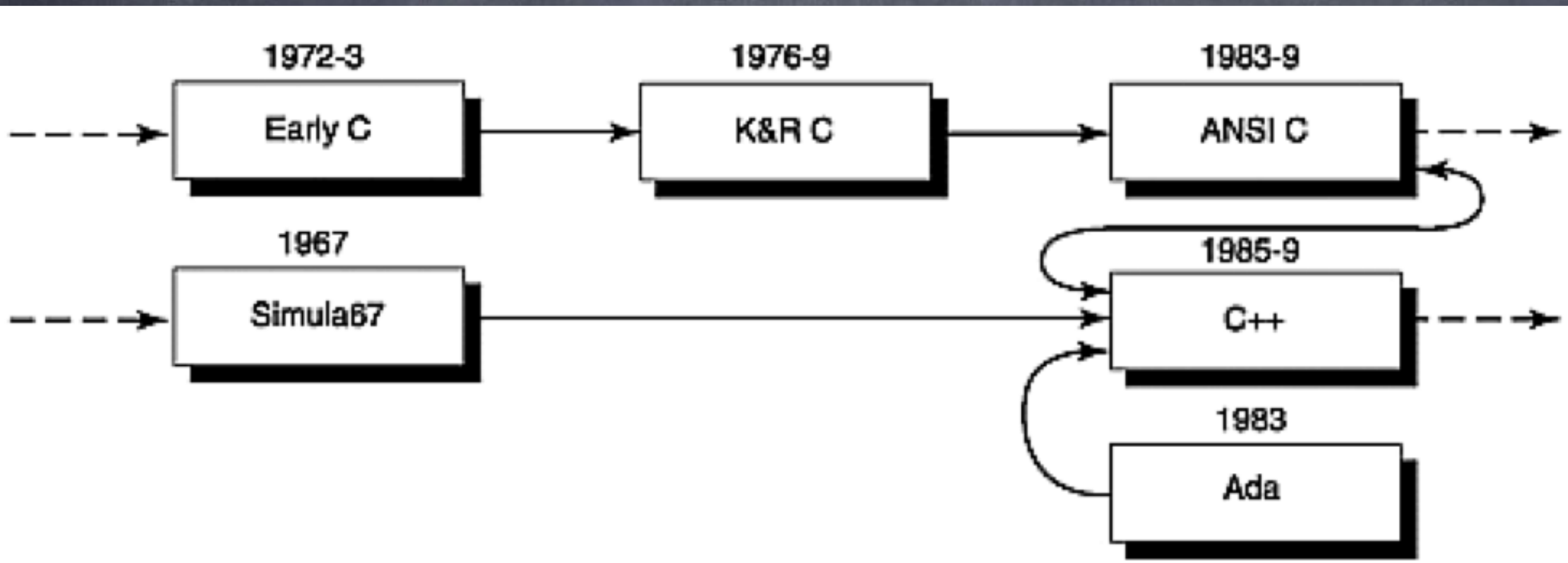
- The type system was added primarily to help the compiler-writer distinguish floats, doubles, and characters from words on the new PDP-11 hardware.
- Many features were put in C for the C compiler-writer's benefit (since C compiler-writers were the chief customers for the first few years).

When was C born?

- the early C

- Features of C that seem to have evolved with the compiler-writer in mind are:
  - Arrays start at 0 rather than 1.
  - The fundamental C types map directly onto underlying hardware. (float, for example)
  - The auto keyword is apparently useless.
  - Array names in expressions "decay" into pointers.
  - Floating-point expressions were expanded to double-length-precision everywhere.

How many Cs?


1972-3
Early C
1976-9
K&R C
1983-9
ANSI C
1967
Simula67
1985-9
C++
1983
Ada

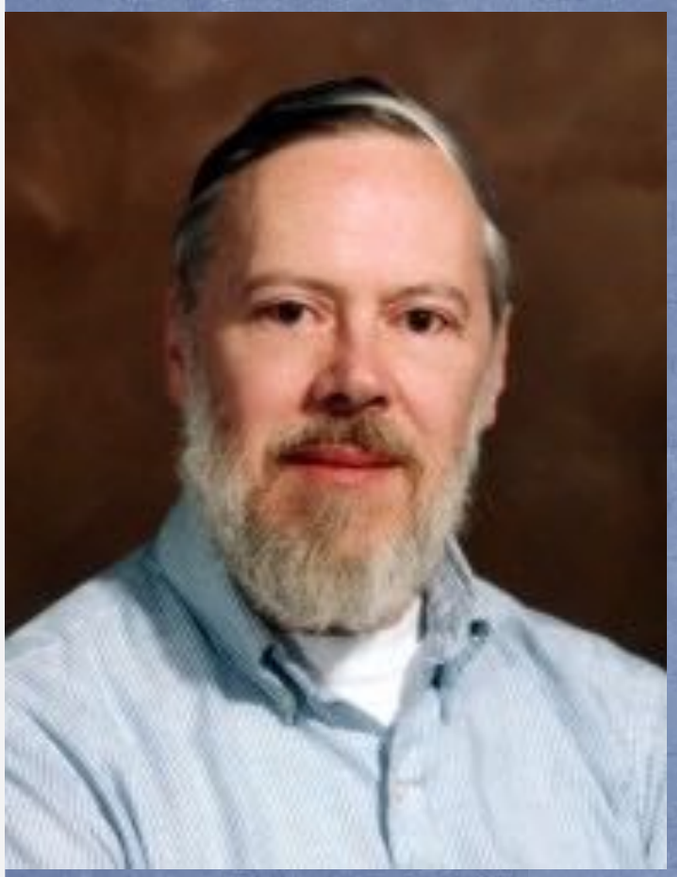
How many Cs?

- By the mid 1970's the language was recognizably the C we know and love today.
- In 1978 Steve Johnson wrote pcc, the portable C compiler. The source was made available outside Bell Labs, and it was very widely ported, forming a common basis for an entire generation of C compilers.

How many Cs?

- K&R C
  - In 1978 the classic C bible, The C Programming Language, written by Brian Kernighan and Dennis Ritchie, was published

Note



C语言和Unix创始人丹尼斯·里奇（Dennis Ritchie）已于2011年10月9日逝世，享年70岁。

里奇于1965年获得哈弗大学物理学博士学位，1967年进入贝尔实验室（Bell Labs），开始了他杰出的职业生涯。1969年，他与肯·汤普逊（Ken Thompson）及其他贝尔实验室人员一起开发Unix操作系统。大约就在那时，他开始研发程序设计语言（即后来的C语言）。1978年，他与汤普逊出版了著名的《C程序设计语言》。

里奇获得的荣誉有：1983年荣获图灵奖，1988年当选美国国家工程院院士以及1999年荣获美国国家技术勋章等等。

How many Cs?

- The Present Day: ANSI C
- In 1983 a C working group formed under the auspices of the American National Standards Institute.

Note

The ANSI C Rationale (only) is available for free by anonymous ftp from the site ftp.uu.net, in directory /doc/standards/ansi/X3.159-1989/.

yet, now online web document available at:

http://www.lysator.liu.se/c/rat/title.html

How many Cs?

Anyone learning or using C should be working with ANSI C, not K&R C

How many Cs?

More about ANSI C

- The ANSI C standard is unique in several interesting ways. It defines the some terms, describing characteristics of an implementation. A knowledge of these terms will aid in understanding what is and isn't acceptable in the language.

How many Cs?

More about ANSI C

- Unportable Code
  - implementation-defined-- The compiler-writer chooses what happens, and has to document it

a rule that is not a constraint

all identifiers declared in the C standard header files are reserved for the implementation, so you may not declare a function called malloc() because a standard header file already has a function of that name. But since this is not a constraint, the rule can be broken, and the compiler doesn't have to warn you!

for example: "interpositioning"

How many Cs?

More about ANSI C

- Portable Code
  - strictly-conforming
    - only uses specified features
    - doesn't exceed any implementation-defined limit
    - has no output that depends on implementation-defined, unspecified, or undefined features .
- conforming-- A conforming program can depend on the nonportable features of an implementation.

How many Cs?

- The Protocol of Prototypes

```
char * strcpy();

char * strcpy(char *dst, const char *src);

char * strcpy(char * , const char * );

char * strcpy(dst, src)
     char *dst, *src;
{ ... }

char * strcpy(char *dst, const char *src)
/* note no semi-colon! */
{ ... }
```

How many Cs?

- assignment operator

b=-3; b= -3;  b-=3;

epsilon=.001; epsilon.=001;

value=!open; value!=open;

Understand ANSI C

about const

- The keyword const doesn't turn a variable into a constant! A symbol with the const qualifier merely means that the symbol cannot be used for assignment.

```
char *cp;
const char *ccp;
ccp = cp;
//cp = ccp;
```

Understand ANSI C

about const

```
const int limit = 10;
const int * limitp = &limit;
int i = 27;
limitp = &i;
//limit = 20;
```

Understand ANSI C

about const

A sales engineer sent the following piece of code into the compiler group at Sun as a test

```
foo(const char **p) { }

main(int argc, char **argv)
{
    foo(argv);
}
```

ANSI C 6.3.2.2:

Each argument shall have a type such that its value may be assigned to an object with the unqualified version of the type of its corresponding parameter.

Understand ANSI C

integral promotions

```
int array[] = { 23, 34, 12, 17, 204, 99, 16 };
#define TOTAL_ELEMENTS (sizeof(array) /
sizeof(array[0]))
main()
{
int d= -1, x;
/* ... */
    if (d <= TOTAL_ELEMENTS-2)
        x = array[d+1];
/* ... */
}
```

Understand ANSI C

Advice on Unsigned Types

- Avoid unnecessary complexity by minimizing your use of unsigned types. Specifically, don't use an unsigned type to represent a quantity just because it will never be negative (e.g., "age" or "national_debt").
- Use a signed type like int and you won't have to worry about boundary cases in the detailed rules for promoting mixed types.
- Only use unsigned types for bitfields or binary masks. Use casts in expressions, to make all the operands signed or unsigned, so the compiler does not have to choose the result type.