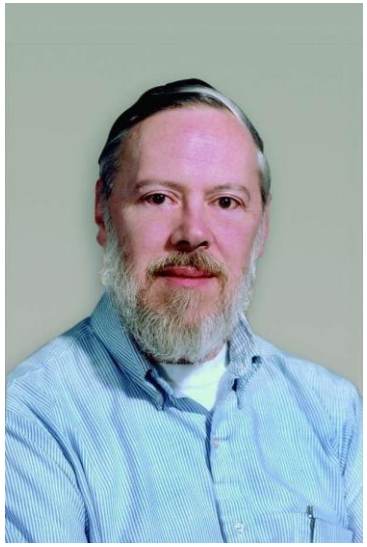
Lecture 1 - Introducing C

Meng-Hsun Tsai
CSIE, NCKU

Origins of C

- C is a by-product of UNIX, developed at Bell Laboratories by Ken Thompson, Dennis Ritchie, and others.
- Thompson designed a small language named B.
- B was based on BCPL, a systems programming language developed in the mid-1960s.

Ken Thompson
(1943-)



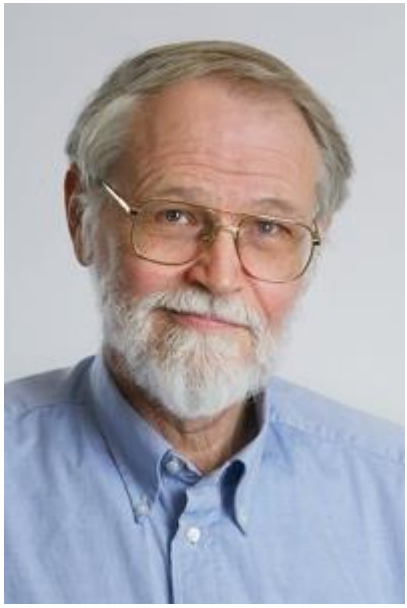
Dennis Ritchie
(1941-2011)

Origins of C (cont.)

- By 1971, Ritchie began to develop an extended version of B.
- He called his language NB (“New B”) at first.
- As the language began to diverge more from B, he changed its name to C.
- The language was stable enough by 1973 that UNIX could be rewritten in C.

Standardization of C

- *K&R C*
  - Described in Kernighan and Ritchie, *The C Programming Language* (1978)
  - De facto standard
- *C89/C90*
  - ANSI standard X3.159-1989 (completed in 1988; formally approved in December 1989)
  - International standard ISO/IEC 9899:1990
- *C99*
  - International standard ISO/IEC 9899:1999
  - Incorporates changes from Amendment 1 (1995)



Brian Kernighan (1942-)

C-Based Languages

- ***C++*** includes all the features of C, but adds classes and other features to support object-oriented programming.
- ***Java*** is based on C++ and therefore inherits many C features.
- ***C#*** is a more recent language derived from C++ and Java.
- ***Perl*** has adopted many of the features of C.

Properties of C

- Low-level
- Small
- Permissive

Strengths of C

- Efficiency (low-level)
- Portability
  - C compilers are small and easily written.
- Power
  - C has large collection of data types and operators.
- Flexibility (small and permissive)
  - C imposes very few restrictions on the use of its features.
- Standard library
- Integration with UNIX

Weaknesses of C

- Programs can be error-prone.
  - Programming mistakes that would be caught in many other languages cannot be detected by a C compiler.
- Programs can be difficult to understand.
  - Programmers who are too clever for their own good can make programs almost impossible to understand.
- Programs can be difficult to modify.
  - Large programs written in C can be hard to change if they haven't been designed with maintenance in mind.

Effective Use of C

- Learn how to avoid pitfalls.
- Use software tools (`lint`, debuggers) to make programs more reliable.
- Take advantage of existing code libraries.
- Adopt a sensible set of coding conventions.
- Avoid “tricks” and overly complex code.
- Stick to the standard.

If you are bored with learning basic C skills…

Take a look at the IOCCC website to see how obfuscating C can be!


安全 | ioccc.org

Logo by winner Matt Zucker

The International Obfuscated C Code Contest

[The judges | **IOCCC home page** | How to enter | FAQ | Mirrors | IOCCC news | People who have won | Winning entries]

The source code for the winners of the 26th IOCCC has been released.

Please see the following news items.

Goals of the Contest

Obfuscate: tr.v. -cated, -cating, -cates.

1. a. To render obscure.
   b. To darken.
2. To confuse: his emotions obfuscated his judgment.

[LLat. obfuscare, to darken : ob(intensive) + Lat. fuscare, to darken < fuscus, dark.] -obfuscation n. obfuscatory adj

The IOCCC:

- To write the most Obscure/Obfuscated C program within the rules.
- To show the importance of programming style, in an ironic way.
- To stress C compilers with unusual code.
- To illustrate some of the subtleties of the C language.
- To provide a safe forum for poor C code. :-)

IOCCC news

Source:http://www.ioccc.org/1986/holloway/holloway.c

IOCCC 1986/holloway

It just prints “Hello, world!”

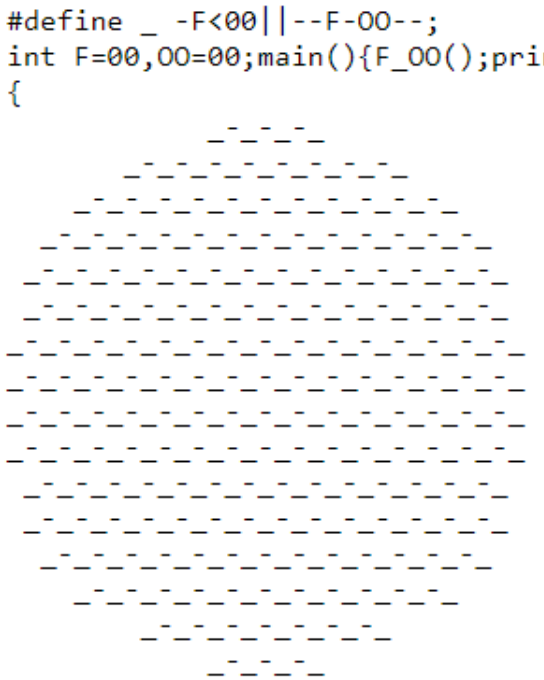
不安全 | ioccc.org/1986/holloway/holloway.c

```
#include "stdio.h"
#define e 3
#define g (e/e)
#define h ((g+e)/2)
#define f (e-g-h)
#define j (e*e-g)
#define k (j-h)
#define l(x) tab2[x]/h
#define m(n,a) ((n&(a))==(a))

long tab1[]={ 989L,5L,26L,0L,88319L,123L,0L,9367L };
int tab2[]={ 4,6,10,14,22,26,34,38,46,58,62,74,82,86 };

main(m1,s) char *s; {
    int a,b,c,d,o[k],n=(int)s;
    if(m1==1){ char b[2*j+f-g]; main(l(h+e)+h+e,b); printf(b); }
    else switch(m1-=h){
        case f:
            a=(b=(c=(d=g)<<g)<<g)<<g;
            return(m(n,a|c)|m(n,b)|m(n,a|d)|m(n,c|d));
        case h:
            for(a=f;a<j;++a)if(tab1[a]&&!(tab1[a]%((long)l(n))))return(a);
        case g:
            if(n<h)return(g);
            if(n<j){n-=g;c='D';o[f]=h;o[g]=f;}
            else{c='\r'-'\b';n-=j-g;o[f]=o[g]=g;}
            if((b=n)>=e)for(b=g<<g;b<n;++b)o[b]=o[b-h]+o[b-g]+c;
            return(o[b-g]%n+k-h);
        default:
            if(m1-=e) main(m1-g+e+h,s+g); else *(s+g)=f;
            for(*s=a=f;a<e;) *s=(*s<<e)|main(h+a++,(char *)m1);
        }
}
```

They both print out the π value.

不安全 | ioccc.org/1988/westley.c

```
#define _ -F<00||--F-OO--;
int F=00,OO=00;main(){F_OO();printf("%1.3f\n",4.*-F/OO/OO);}F_OO()
{
```



```
}
```

Source:http://www.ioccc.org/1988/westley.c

不安全 | ioccc.org/1989/roemer.c

```
                                                                  char
                                                          _3141592654[3141
      ],__3141[3141];_314159[31415],_3141[31415];main(){register char*
    _3_141,*_3_1415, *_3__1415; register int _314,_31415,__31415,*_31,
  _3_14159,__3_1415;*_3141592654=__31415=2,_3141592654[0][_3141592654
 -1]=1[__3141]=5;__3_1415=1;do{_3_14159=_314=0,__31415++;for( _31415
=0;_31415<(3,14-4)*__31415;_31415++)_31415[_3141]=_314159[_31415]= -
;_3141[*_314159=_3_14159]=_314;_3_141=_3141592654+__3_1415;_3_1415=
_3_1415    +__3141;for                (_31415 = 3141-
         __3_1415  ;                  _31415;_31415--
         ,_3_141 ++,                  _3_1415++){_314
         +=_314<<2 ;                  _314<<=1;_314+=
        *_3_1415;_31                   =_314159+_314;
        if(!(*_31+1)                   )* _31 =_314 /
        __31415,_314                   [_3141]=_314 %
        __31415 ;* (                   _3__1415=_3_141
       )+= *_3_1415                     = *_31;while(*
       _3__1415 >=                      31415/3141 ) *
       _3__1415+= -                     10,(*--_3__1415
      )++;_314=_314                     [_3141]; if ( !
      _3_14159 && *                     _3_1415)_3_14159
      =1,__3_1415 =                     3141-_31415;}if(
      _314+(__31415                      >>1)>=__31415 )
      while ( ++ *                       _3_141==3141/314
     )*_3_141--=0                        ;}while(_3_14159
     ) ; { char *                        __3_14= "3.1415";
     write((3,1),                        (--*__3_14,__3_14
     ),(_3_14159                          ++,++_3_14159))+
    3.1415926; }                          for ( _31415 = 1;
   _31415<3141-                           1;_31415++)write(
  31415% 314-(                            3,14),_3141592654[
 _31415    ] +                           "0123456789","314"
 [ 3]+1)-_314;                           puts((*_3141592654=0
,_3141592654))                            ;_314= *"3.141592";}
```

Source:http://www.ioccc.org/1989/roemer.c

IOCCC 1998/banks

It is a flight simulator in 1536 bytes! (X11 needed)

Source:http://www.ioccc.org/1998/banks.c

Source:https://www.tiobe.com/tiobe-index/

Does it deserve to learn C?

Sep 2020	Sep 2019	Change	Programming Language	Ratings	Change
1	2	↑	C	15.95%	+0.74%
2	1	↓	Java	13.48%	-3.18%
3	3		Python	10.47%	+0.59%
4	4		C++	7.11%	+1.48%
5	5		C#	4.58%	+1.18%
6	6		Visual Basic	4.12%	+0.83%
7	7		JavaScript	2.54%	+0.41%
8	9	↑	PHP	2.49%	+0.62%
9	19	⇈	R	2.37%	+1.33%
10	8	↓	SQL	1.76%	-0.19%
11	14	↑	Go	1.46%	+0.24%
12	16	⇈	Swift	1.38%	+0.28%
13	20	⇈	Perl	1.30%	+0.26%

Source:https://www.tiobe.com/tiobe-index/

Does it deserve to learn C? (cont.)

Very Long Term History

To see the bigger picture, please find below the positions of the top 10 programming languages of many years back. Please note that these are *average* positions for a period of 12 months.

Programming Language	2020	2015	2010	2005	2000	1995	1990	1985
Java	1	2	1	2	3	-	-	-
C	2	1	2	1	1	2	1	1
Python	3	7	6	6	20	20	-	-
C++	4	3	3	3	2	1	2	9
C#	5	5	5	7	9	-	-	-
JavaScript	6	8	8	10	6	-	-	-
PHP	7	6	4	5	19	-	-	-
SQL	8	-	-	-	-	-	-	-
Swift	9	16	-	-	-	-	-	-
R	10	12	52	-	-	-	-	-