Learning C in 2 Hours

D. Thiebaut

THE **C** PROGRAMMING LANGUAGE

Brian W. Kernighan • Dennis M. Ritchie

PRENTICE HALL SOFTWARE SERIES

- Dennis Ritchie

- 1969 to 1973

- AT&T Bell Labs

- Close to Assembly

- Unix

- Standard

- Many languages based on C.

Inspiration

- C Language Tutorial from Drexel U.
  http://www.physics.drexel.edu/courses/Comp_Phys/General/C_basics/

Hello World!

- Library

- Strings

- Block-structured language

- main()

```c
#include <stdio.h>

void main() {
    printf("\nHello World\n");
}
```

Compiling on Aurora

- gcc Gnu compiler

- `man` gcc for help

```
[~/handout]$ gcc hello.c
[~/handout]$ ./a.out

Hello World
```

```
[~/handout]$ gcc -o hello hello.c
[~/handout]$ ./hello

Hello World
```
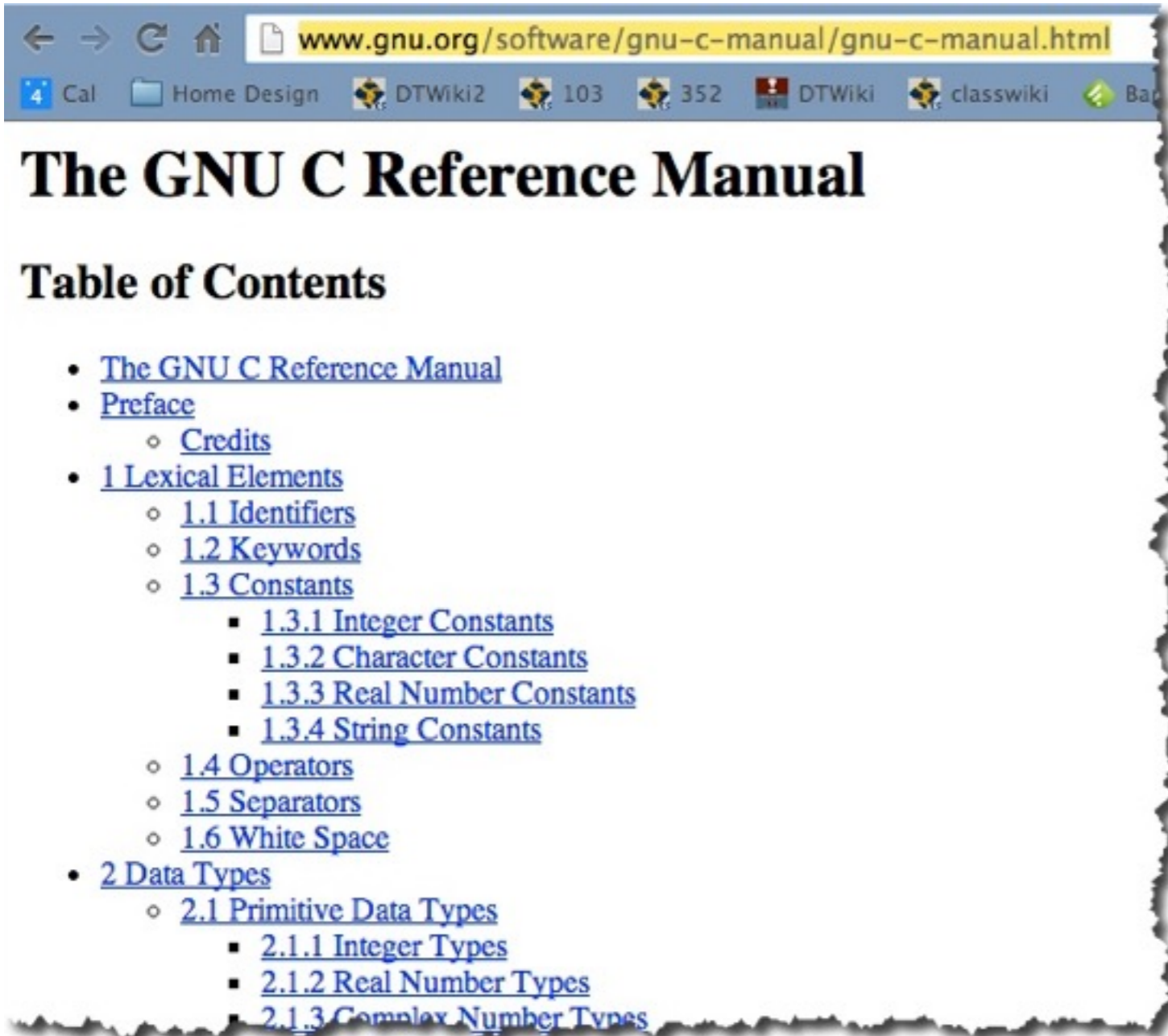
Files

```
[~/handout]$ ls -l
total 28
-rwx------ 1 352a 352a 6583 Oct  6 16:41 a.out*
-rwx------ 1 352a 352a 6583 Oct  6 16:48 hello*
-rw------- 1 352a 352a   66 Oct  6 16:41 hello.c
-rw------- 1 352a 352a   67 Oct  6 16:39 hello.c~
```

Good Reference

The GNU C Reference Manual

Table of Contents

- http://www.gnu.org/software/gnu-c-manual/gnu-c-manual.html

Printing

- printf("string with %-operators", list of vars);

    - %d int

    - %f float

    - %s string

Variables

- Simple types

- No strings!

- No booleans (only 0 for *false* and !0 for *true*)

- No classes, no objects!

```
int     -> integer variable
short   -> short integer
long    -> long integer
float   -> single precision real (floating point) variable
double  -> double precision real (floating point) variable
char    -> character variable (single byte)
```

Comments

```c
/*
programName.c
author

This is the header
*/
#include <stdio.h>
#include <string.h>

void main() {
    // line comment
    print( "Hello!\n" )
}
```

Strings

```c
#include <stdio.h>
#include <string.h>

void main() {
  char hello[] = "hello";
  char world[] = "world!";
  char sentence[100] = "";

  strcpy( sentence, hello );   // sentence <- "hello"
  strcat( sentence, " " );     // sentence <- "hello "
  strcat( sentence, world );   // sentence <- "hello world!"

  printf( "sentence = %s\n", sentence );
}
```

```
[~/handout]$ gcc strings2.c

sentence = hello world!
[~/handout]
```

Strings end with '\0'

```
#include <stdio.h>
#include <string.h>

void main() {
  char sentence[100] = "Hello world!";

  printf( "sentence = %s\n", sentence );
  sentence[5] = '\0';
  printf( "sentence = %s\n", sentence );
}
```

```
~/handout]$ a.out
sentence = Hello world!
sentence = Hello
[~/handout]$
```

Exercise

```c
#include <stdio.h>
#includd <string.h>

void main() {
  int a   = 3;
  int b   = 5;
  int c   = 0;
  char firstName[] = "your first name here";
  char lastName[] = "your last name here";
  char fullName[100];
  ...
}
```

- make the program store the sum of a and b into c, and then print your full name and the value in c. Also, make it output the number of characters in your full name (it must count the number of chars using a string function (strlen))

Solution

```c
#include <stdio.h>
#include <string.h>

void main() {
  int a = 3;
  int b = 5;
  int c = 0;
  char firstName[] = "Mickey";
  char lastName[] = "Mouse";
  char fullName[100];

  c = a + b;
  strcpy( fullName, firstName );
  strcat( fullName, " " );
  strcat( fullName, lastName );

  printf( "\nc = %d  Full name = %s\n\n", c, fullName );
}
```

For-Loops

No "int" declaration!!!

```c
#include <stdio.h>

void main() {
   int i;
   int sum = 0;

   // compute the sum of all the numbers from 1 to 100
   for ( i=1; i<=100; i++ ) {
     sum += i;
   }


   printf( "\nsum = %d\n\n", sum );
}
```

While-Loops

```c
#include <stdio.h>

void main() {
  int i;
  int sum = 0;

  // compute the sum of all the numbers from 1 to 100
  i = 1;
  while ( i<=100 ) {
    sum += i;     // could have also used i++
    i += 1;
  }

  printf( "\nsum = %d\n\n", sum );
}
```

Infinite Loops

```
#include <stdio.h>

void main() {

    while ( 1 ) {
        printf( "hello!\n" );
    }
}
```

```
#include <stdio.h>

void main() {

    for ( ;; ) {
        printf( "hello!\n" );
    }
}
```

Exercise

- Write a program that writes your full name underlined (line of dashes below). The underline length must be computed and the number of characters equal to the number of characters in your full name.

  Hints: strlen() returns # of chars

```
man strlen
```

Symbolic Constants

```c
#include <stdio.h>

#define NAME       "Mickey"
#define HEIGHT     5
#define YEARBORN   1928


void main() {
  printf( "%s is %d inches high, and was created in %d\n\n",
          NAME, HEIGHT, YEARBORN );
}
```

Conditionals

```c
#include <stdio.h>

void main() {
  int a = 5;
  int b = 3;
  int c = 7;

  if ( a <= b && a <= c )
    printf( "%d is the smallest\n\n", a );
  else if ( b <= a && b <= c )
    printf( "%d is the smallest\n\n", b );
  else
    printf( "%d is the smallest\n\n", c );
}
```

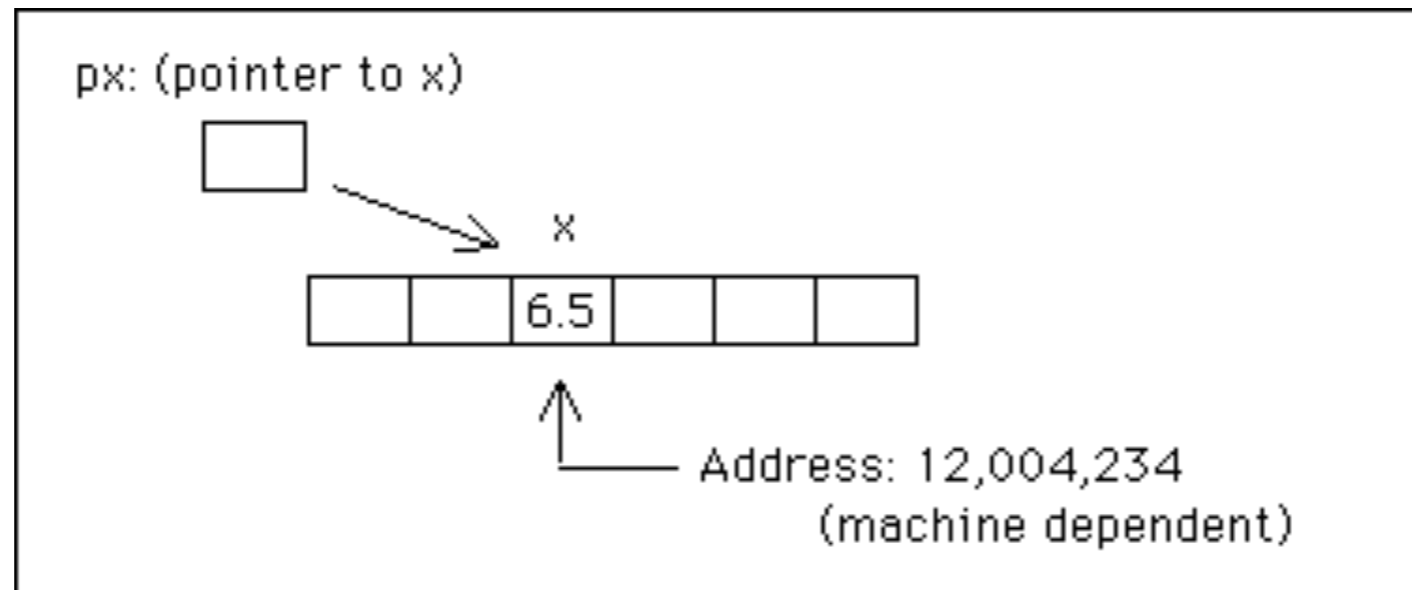Computer Science

Conditionals (cont'd)

```
switch ( ordinal_expression ) {
  case ordinal_value: {
    // ...
    break;
  }
  case ordinal_value: {
    // ...
    break;
  }
  default: {
    // ...
  }
}
```

Pointers

Concept


```
float x = 6.5;
float* px = &x;
```

Example: Initialize an Array

```c
#include <stdio.h>
#define SIZE 10

int  main() {
    float A[SIZE];
    float* p;
    int i;

    p = A;
    for ( i=0; i<SIZE; i++ ) {
        *p = i;
        p++;
    }

    p = A;
    for ( i=0; i<SIZE; i++ ) {
        printf( "p=%p A[%d] = %1.2f *p = %1.2f\n",
                p, i, A[i], *p );
        p = p + 1;
    }
}
```

```
a.out
p=0x7fff88d54560 A[0] = 0.00 *p = 0.00
p=0x7fff88d54564 A[1] = 1.00 *p = 1.00
p=0x7fff88d54568 A[2] = 2.00 *p = 2.00
p=0x7fff88d5456c A[3] = 3.00 *p = 3.00
p=0x7fff88d54570 A[4] = 4.00 *p = 4.00
p=0x7fff88d54574 A[5] = 5.00 *p = 5.00
p=0x7fff88d54578 A[6] = 6.00 *p = 6.00
p=0x7fff88d5457c A[7] = 7.00 *p = 7.00
p=0x7fff88d54580 A[8] = 8.00 *p = 8.00
p=0x7fff88d54584 A[9] = 9.00 *p = 9.00
```

Arrays
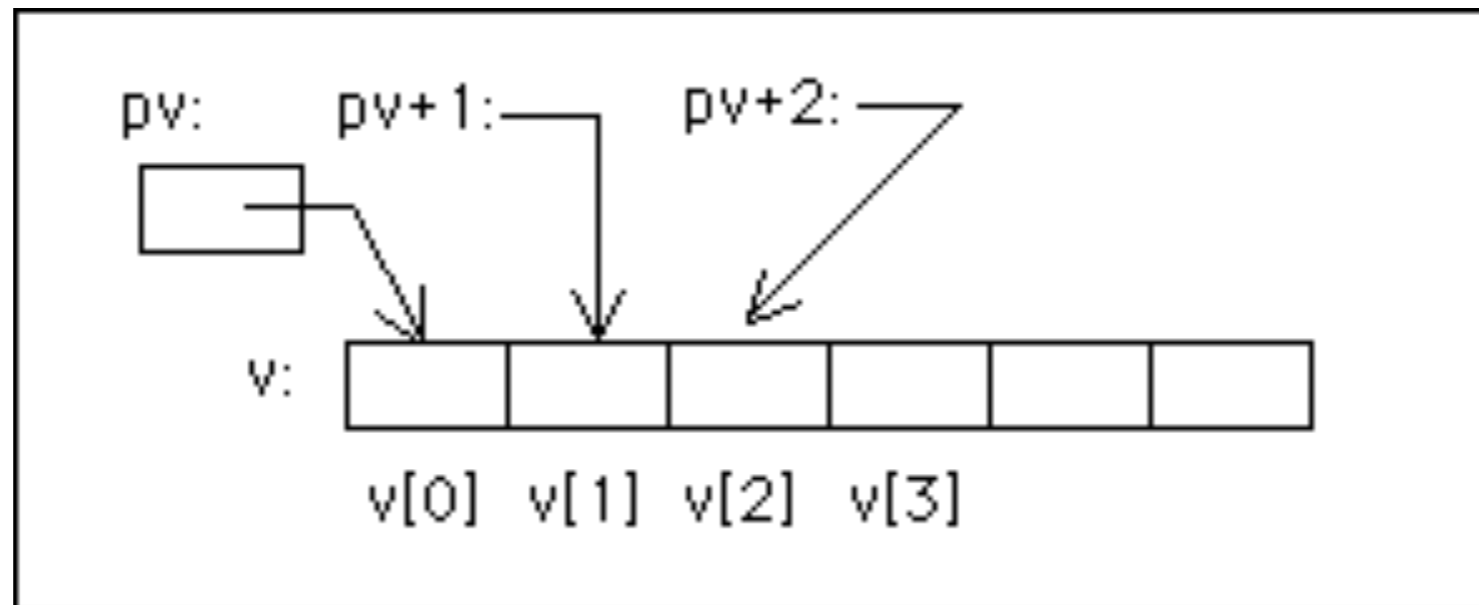

```
TYPE v[DIM];
TYPE* pv;

pv = v;
```

Arrays

- The name of an array is a pointer to the first cell of the array.

```
char name[DIM];
```

- name is the same as &(name[0])

* and &

- * has two meanings, depending on context
  - "Pointer to"
  - "Contents of"
- & means "the address of"

* and &

```
int A[DIM];
int* p = A;          // "int pointer p"
int *q = A;          // "int pointer q"

*p = 3;              // what p is pointing to gets 3
```

Functions

- Functions are always declared before they are used

- Functions can return values of simple types (int, char, floats), and *even* pointers.

- Functions get parameters of simple types, and pointers.

- Passing by value is automatic. Passing by reference requires passing a pointer.

Functions

```c
#include <stdio.h>

int sum( int a, int b ) {
    return a+b;
}


void main() {
    int x = 10;
    int y = 20;
    int z;

    z = sum( x, y );
    printf( "z = %d\n", z );

    z = sum( 3, 8 );
    printf( "z = %d\n", z );

    printf( "sum( 11, 22) = %d\n", sum( 11, 22 ) );
}
```

```
z = 30
z = 11
sum( 11, 22) = 33
```

Functions

```c
#include <stdio.h>

void sum2( int a, int b, int  c ) {
    c = a+b;
}


void main() {
    int x = 10;
    int y = 20;
    int z;

    sum2( x, y,    z );
    printf( "z = %d\n", z );

    sum2( 3, 8,    x );
    printf( "x = %d\n", x );
}
```

```
z = 30
x = 11
```

Input

```c
#include <stdio.h>

void main() {
    int age;
    float myPi;
    char name[80];

    printf( "Enter your name, please: " );
    gets( name ); // could be dangerous
                  // if more than 80 chars entered

    printf( "Enter your age: " );
    scanf( "%d", &age );

    printf( "Enter your version of pi: " );
    scanf( "%f", &myPi );

    printf( "%s is %d years old, and thinks pi is %1.10f\n\n",
            name, age, myPi );
}
```

Input (cont'd)

```
a.out
Enter your name, please: Dominique
Enter your age: 21
Enter your version of pi: 3.14159
Dominique is 21 years old, and thinks pi is 3.1415901184
```

Exercises

```c
#include <stdio.h>
#include <stdlib.h>

#define   N 10

// functions go here...

void main() {
    int A[N] = { 3, 2, 1, 0, 6, 5, 9, 8, 7 };

    // your code goes here
}
```

- Write a C program (no functions) that computes the smallest, the largest, and the sum of all the ints in A.

- Write another program that does the same thing but uses functions. Only passing by value is allowed.

Exercises

- Write another program that does the same thing but uses functions, but this time passing by reference is required.

Dynamic Variables

Malloc

```c
#include <stdio.h>
#include <stdlib.h>

void main() {
    int *A, N, i, smallest;

    printf( "How many ints? " );
    scanf( "%d", &N );
    A = (int *) malloc( N * sizeof( int ) );

    for ( i=0; i<N; i++ ) {
        printf( "> " );
        scanf( "%d", &(A[i]) );
    }

    smallest = A[0];
    for ( i=1; i<N; i++ )
        if ( A[i] < smallest )
            smallest = A[i];
    free( A );

    printf( "The smallest = %d\n", smallest );
}
```

Exercise

- Same as previous exercises (finding the smallest, largest, sum of some array), but this time you allocate the array dynamically from the user input, i.e. ask the user for number of ints, then the ints.

File I/O output

```c
#include <stdio.h>

void main() {
    FILE *fp;
    int i;
    char name[] = "Smith College";

    fp = fopen("hello.txt", "w");  // open file for writing

    fprintf(fp, "\nHello " );       // write constant string
    fprintf(fp, "%s\n\n", name );  // write string

    fclose(fp);                     // close file
}
```

File I/O: Reading Text

```c
#include <stdio.h>

void main() {
    FILE *fp;
    char line[80];

    fp = fopen( "fgets2.c", "r" );    // open file for reading

    while ( !feof( fp ) ) {            // while not eof
        fgets( line, 80, fp );        // get at most 80 chars
        if ( feof( fp ) )             // if eof reached stop
            break;
        line[79] = '\0';              // truncate line to be safe
        printf( "%s", line );         // print it
    }

    close( fp );                      // close file
}
```

File I/O: Input Numbers

```
[~handout] cat fileOfInts.txt
4
1 2 3
4 5 6
7 8 9
10 11 12
```

SMITH COLLEGE

File I/O: Input Numbers

```c
#include <stdio.h>

void main() {
    FILE *fp;
    char line[80];
    int N, n1, n2, n3;

    fp = fopen( "fileOfInts.txt", "r" ); // 1st number is # of lines
                                         // then 3 ints per line

    if ( feof( fp ) ) {
        printf( "Empty file!\n\n" );
        return;
    }

    // get the number of lines
    fscanf( fp, "%d", &N );

    while ( !feof( fp ) ) {
        fscanf( fp, "%d %d %d", &n1, &n2, &n3 );
        if ( feof( fp ) )
            break;
        printf( "%d, %d, %d\n", n1, n2, n3 );
    }

    close( fp );
}
```

File I/O: Input Numbers

```
[~handout] a.out
1, 2, 3
4, 5, 6
7, 8, 9
10, 11, 12
```

Dominique Thiebaut

SMITH COLLEGE

Exercise

- Same as last exercise, but this time the array is stored in a text file. The first number on the first line (only one number on first line) is the dimension of the array. Then there is one number per line. Program stores all the numbers in the array and prints the smallest.