C Programming
A Modern Approach

K. N. King,
C Programming
A Modern Approach,
W. W. Norton & Company, 1996.

Original Notes by Raj Sunderraman
Converted to presentation and updated by
Michael Weeks

Note About Coverage

- This class has only part of the semester dedicated to C
- You already know Java, which has similar syntax
- You should be able to read the book quickly

Similarities of C to Java

- /* Comments */
- Variable declarations
- If / else statements
- For loops
- While loops
- Function definitions (like methods)
- Main function starts program

Differences between C and Java

- C does not have objects
  - There are “struct”ures
  - But data are not tied to methods
- C is a functional programming language
- C allows pointer manipulation
- Input / Output with C
  - Output with printf function
  - Input with scanf function

Variable Type

C has the following simple data types:

C type	Size (bytes)	Lower bound	Upper bound
char	1	—	—
unsigned char	1	0	255
short int	2	−32768	+32767
unsigned short int	2	0	65536
(long) int	4	-2^{31}	$+2^{31}-1$
float	4	$-3.2 \times 10^{\pm 38}$	$+3.2 \times 10^{\pm 38}$
double	8	$-1.7 \times 10^{\pm 308}$	$+1.7 \times 10^{\pm 308}$

Java has the following simple data types:

Primitive type	Size	Minimum	Maximum	Wrapper type
boolean	1-bit	–	–	**Boolean**
char	16-bit	Unicode 0	Unicode 2^{16}- 1	**Character**
byte	8-bit	-128	+127	**Byte**[1]
short	16-bit	-2^{15}	$+2^{15}-1$	**Short**[1]
int	32-bit	-2^{31}	$+2^{31}-1$	**Integer**
long	64-bit	-2^{63}	$+2^{63}-1$	**Long**
float	32-bit	IEEE754	IEEE754	**Float**
double	64-bit	IEEE754	IEEE754	**Double**

In C:

```
char ch;

ch = ‘a’;
ch = ‘A’;
ch = ‘0’;
ch = ‘ ‘;


char ch;
int I;
i = ‘a’
ch = 65;
ch = ch + 1;
ch++;

if (‘a’<=ch &&
ch<=‘z’)
ch=ch-’a’+’A’
```

Data Types

- char, int, float, double
- long int (long), short int (short), long double
- signed char, signed int
- unsigned char, unsigned int
  - 1234L is long integer,
  - 1234 is integer,
  - 12.34 is float,
  - 12.34L is long float

Data Types

- 'a', '\t', '\n', '\0', etc. are character constants
- strings: character arrays
  - (see <string.h> for string functions)
  - "I am a string"
  - always null terminated.
  - 'x' is different from "x"

Type Conversions

- narrower types are converted into wider types
  - f + i int i converted to float
- characters <---> integers
- <ctype.h> library contains conversion functions, e.g.:
  - tolower(c) isdigit(c) etc.
- Boolean values:
  - true : >= 1 false: 0

```
int atoi(char s[]) {
   int i, n=0;
   for (i=0; s[i] >= '0'
&& s[i] <= '9'; i++)
     n = 10*n + (s[i]-'0');

   return n;
}
```

Pointers and Arrays

- Pointer variable contains the address of another variable
  - unary operator & applied to variables gives the address of variable
  - unary operator * applied to pointer accesses the variable pointer points to
- char c; p = &c;
  - address of c is assigned to variable p

Pointers and Arrays

```
int x=1, y=2, z[10];
int *p;  /* p points to an integer */

p = &x;  /* Set p to x's address */
y = *p;  /* Get value of p, store in y */

*p = 0;      /* Set p's value to 0 */
p = &z[0];  /* Set p to z[0]'s address */
```

Pointer Example

```
$ cat ptr_example.c
#include <stdio.h>

int main() {
    int x=1;
    int *p;  /* p points to an integer */

    p = &x;  /* Set p to x's address */
    printf(" x is %d\n", x);

    *p = 0;  /* Set p's value to 0 */
    printf(" x now is %d\n", x);

    return 0;
}

$ gcc ptr_example.c -o ptr_example
$ ./ptr_example
 x is 1
 x now is 0
```

Using pointers to achieve Call-By-Reference effect

- Pass the address of a variable
- Alter the value

```
void swap (int *px, int *py) {
  int temp;
  temp = *px;
  *px = *py;
  *py = temp;
}

int a=10,b=20;
swap(&a,&b);
```

Pointers and Arrays

```
int a[10], *pa, x;

pa = &a[0];
x = *pa;
x = *(pa+1);
x = *(pa+5);
pa = a; /* same as pa = &a[0];
```

Main difference between arrays and pointers

- Even though both contain addresses:
- Array name is not a variable; so
- pa = a; pa++ OK
- a = pa; a++; NOT OK
- When an array name is passed as a parameter to a function:
  - Actually the address of the first element is passed
  - Arrays are always passed as pointers

Finding the length of a String

```
int strlen(char *s) {  /* One way */
  int n;

  for (n=0; *s!='\0'; s++)
    n++;

  return n;
}


int strlen(char s[]) {  /* Another possibility */

  int n;

  for (n=0; s[n]!='\0';n++);

  return n;
}
```

Strings

- Array of characters
- Example: string copy

```
void strcpy(char *s, char *t) {
  int i=0;
  while ((s[i]= t[i]) != '\0')
    i++;
}

/* OR: */
while ((*s = *t) != '\0') {
  s++; t++;
}
```

Scope Rules

- Automatic/Local Variables
  - Declared at the beginning of functions
  - Scope is the function body
- External/Global Variables
  - Declared outside functions
  - Scope is from the point where they are declared until end of file (unless prefixed by extern)

Scope Rules

- Static Variables: use static prefix on functions and variable declarations to limit scope
  - static prefix on external variables will limit scope to the rest of the source file (not accessible in other files)
  - static prefix on functions will make them invisible to other files
  - static prefix on internal variables will create permanent private storage; retained even upon function exit

Scope Rules

- Variables can be declared within blocks too
  - scope is until end of the block

Bitwise Operations

- Applied to char, int, short, long
  - And &
  - Or |
  - Exclusive Or ^
  - Left-shift <<
  - Right-shift >>
  - one's complement ~

Example: Bit Count

```
/*
   count the 1 bits in a number
   e.g. bitcount(0x45) (01000101 binary) returns 3
*/

int bitcount (unsigned int x) {
   int b;

   for (b=0; x != 0; x = x >> 1)
      if (x & 01)     /* octal 1 = 000000001 */
          b++;

   return b;
}
```

Conditional Expressions

- Conditional expressions
- expr1? expr2:expr3;
- if expr1 is true then expr2 else expr3

```
for (i=0; i<n; i++)
   printf("%6d %c",a[i],(i%10==9||i==(n-1))?'\n':' ');
```

Control Flow

- **blocks: { ... }**
- **if expr stmt;**
- **if expr stmt1 else stmt2;**
- **switch expr {case ... default }**
- **while expr stmt;**
- **for (expr1;expr2;expr3) stmt;**
- **do stmt while expr;**
- **break; continue (only for loops);**
- **goto label;**

Hello, World

```
#include <stdio.h>
/* Standard I/O library */

/* Function main with no arguments */
int main () {
  /* call to printf function */
  printf("Hello, World!\n");

  /* return SUCCESS = 1 */
  return 1;
}

% gcc -o hello hello.c
% ./hello
Hello, World!
%
```

Celsius vs Fahrenheit table (in steps of 20F)

- C = (5/9)*(F-32);

```
#include <stdio.h>
int main() {
  int fahr, celsius, lower, upper, step;
  lower = 0;
  upper = 300;
  step = 20;
  fahr = lower;
  while (fahr <= upper) {
    celsius = 5 * (fahr - 32) / 9;
    printf("%d\t%d\n",fahr, celsius);
    fahr += step;
  }
  return 1;
}
```

```
$ ./CelsiusFahrenheitTa
0       -17
20      -6
40      4
60      15
80      26
100     37
120     48
140     60
160     71
180     82
200     93
220     104
240     115
260     126
280     137
300     148
```

Celsius vs Fahrenheit table Remarks

- 5/9 = 0
- Primitive data types: int, float, char, short, long, double
- Integer arithmetic: 0F = 17C instead of 17.8C
- %d, %3d, %6d etc for formatting integers
- \n newline
- \t tab

New Version Using Float

```
#include <stdio.h>
int main() {
  float fahr, celsius;
  int lower, upper, step;
  lower = 0;
  upper = 300;
  step = 20;
  fahr = lower;
  while (fahr <= upper) {
    celsius = (5.0 / 9.0) * (fahr - 32.0);
    printf("%3.0f %6.1f \n", fahr, celsius);
    fahr += step;
  }
  return 1;
}
```

Results:

```
  0  -17.8
 20   -6.7
 40    4.4
 60   15.6
 80   26.7
100   37.8
120   48.9
140   60.0
160   71.1
180   82.2
200   93.3
220  104.4
240  115.6
260  126.7
280  137.8
300  148.9
```

New Version Using Float Remarks

- %6.2f 6 wide; 2 after decimal
- 5.0/9.0 = 0.555556
- Float has 32 bits
- Double has 64 bits
- Long Double has 80 to 128 bits
  - Depends on computer

Version 3 with “for” loop

```
#include <stdio.h>

int main() {
  int fahr;

  for (fahr=0; fahr <= 300; fahr += 20)

printf("%3d %6.1f \n", fahr, (5.0 / 9.0) * (fahr –
32.0));

  return 1;
}
```

Results:

```
0  -17.8
 20   -6.7
 40    4.4
 60   15.6
 80   26.7
100   37.8
120   48.9
140   60.0
160   71.1
180   82.2
200   93.3
220  104.4
240  115.6
260  126.7
280  137.8
300  148.9
```

Version 4 with Symbolic Constants

```
#include <stdio.h>

#define LOWER 0
#define UPPER 300
#define STEP 20

int main() {
  int fahr;

  for (fahr=LOWER; fahr <= UPPER; fahr += STEP)
    printf("%3d %6.1f \n", fahr,
          (5.0 / 9.0) * (fahr - 32.0));

  return 1;
}
```

Results:

```
0  -17.8
 20   -6.7
 40    4.4
 60   15.6
 80   26.7
100   37.8
120   48.9
140   60.0
160   71.1
180   82.2
200   93.3
220  104.4
240  115.6
260  126.7
280  137.8
300  148.9
```

Character I/O

- We use “int” instead of “char” below
  - Input functions also return End-of-file (-1) and error conditions
  - Use feof() and ferror() to tell the difference
- int c;
- c = getchar();
- putchar(c);

File Copying

```
#include <stdio.h>

int main() {
  int c;

  c = getchar();
  while (c != EOF) {
    putchar(c);
    c = getchar();
  }

  return 0;
}
```

File Copying (Simpler Version)

- c= getchar() != 0 is equivalent to
  c = (getchar() != EOF)
- Results in c value of 0 (false) or 1 (true)

```
#include <stdio.h>
int main() {
  int c;

  c = getchar();
  while ((c = getchar())!= EOF)
    putchar(c);

  return 0;
}
```

Counting Characters

- Remarks: nc++, ++nc, --nc, nc--
- %ld for long integer

```
#include <stdio.h>
int main () {
  long nc = 0;
  while (getchar() != EOF)
    nc++;
  printf("%ld\n",nc);
}
```

```
#include <stdio.h>
int main () {
  long nc;
  for (nc=0;getchar() != EOF;nc++);
  printf("%ld\n",nc);
}
```

Counting Lines

```
#include <stdio.h>

int main () {
  int c, nl=0;

  while ((c = getchar()) != EOF)
    if (c == '\n')
      nl++;

  printf("%d\n",nl);
}
```

Counting Words

```
#include <stdio.h>
#define IN 1
#define OUT 0
int main () {
  int c, nl, nw, nc, state;

  state = OUT;
  nl = nw = nc = 0;
  while ((c = getchar()) != EOF) {
    ++nc;
    if (c == '\n')
      nl++;
    if (c == ' ' || c == '\n' || c == '\t')
      state = OUT;
    else if (state == OUT) {
          state = IN;
          ++nw;
        }
  }
  printf("%d %d %d\n",nc, nw, nl);  }
```

Notes about Word Count

- Short-circuit evaluation of || and &&
- nw++ at the beginning of a word
- use state variable to indicate inside or outside a word

Arrays

- Count #occurrences of **each digit, blank, tab, newline, other characters**

```
#include <stdio.h>
int main() {
  int c, i, nwhite, nother;
  int ndigit[10];

  nwhite = nother = 0;
  for (i=0; i<10; ++i)
    ndigit[i] = 0;
  while ((c = getchar()) != EOF)
    if (c > '0' && c < '9')
      ++ndigit[c-'0'];
    else if (c == ' ' || c == '\n' || c == '\t')
          ++nwhite;
        else
          ++nother;
  printf("digits = ");
  for (i=0; i<10; ++i)
    printf("%d ",ndigit[i]);
  printf(", white space = %d, other =
%d\n",nwhite, nother);
}
```

Functions

```
#include <stdio.h>

int power(int, int);          /* function prototype */

int main() {
  int i;
  for (i = 0; i < 10; i++)
    printf("%d %d %d\n", i, power(2,i), power(-3,i));
}

int power(int base, int n) {
  int p;
  for (p = 1; n > 0; --n)
    p = p * base;
  return p;
}
```

Results:

```
0 1 1
1 2 -3
2 4 9
3 8 -27
4 16 81
5 32 -243
6 64 729
7 128 -2187
8 256 6561
9 512 -19683
```

Functions

- Call by value mechanism
- Change in n's value not reflected in main
- Using pointers, we can achieve Call by reference effect.

Functions

```
/* Character Arrays
   Read a set of text lines and print the longest
*/
#include <stdio.h>
#define MAXLINE 1000

int getline(char [],int);
void copy(char [], char []);

int main() {
  int len, max;
  char line[MAXLINE], longest[MAXLINE];
  max = 0;
  while ((len = getline(line,MAXLINE)) > 0)
    if (len > max) {
      max = len;
      copy(longest,line);
    }
  if (max > 0)
    printf("%s",longest);
}
```

Getline and Copy

```
int getline(char s[], int limit) {
  int c, i;

  for (i=0; i<(limit - 1) && (c=getchar())!=EOF
       && c != '\n'; i++)
    s[i] = c;
  if (c == '\n') {
    s[i] = c;
    i++;
  }
  s[i] = '\0';
  return i;
}

void copy(char to[], char from[]) {
  int i=0;

  while ((to[i] = from[i]) != '\0')
    i++;
}
```

External (Global) Variables

- All variables declared so far are local to the functions
- External or Global variables are accessible to all functions
- These are defined once outside of functions
- Must be defined once more within each function which is going to use them
- Previous program rewritten with external variables

New Version with Externals

```
/* Longest line program with line, longest, max as external variables */
#include <stdio.h>
#define MAXLINE 1000
int max;
char line[MAXLINE];
char longest[MAXLINE];
int getline(void);      /* no parameters */
void copy(void);        /* no parameters */
int main() {
  int len;
  extern int max;
  extern char longest[];
  max = 0;
  while ((len = getline()) > 0)
    if (len > max) {
      max = len;
      copy();
    }
  if (max > 0)
    printf("%s",longest);
}
```

Getline and Copy with Externals

```
int getline(void) {
  int c, i;
  extern char line[];
  for (i=0; i<(MAXLINE - 1) && (c=getchar())!=EOF && c != '\n'; i++)
    line[i] = c;
  if (c == '\n') {
    line[i] = c;
    i++;
  }
  line[i] = '\0';
  return i;
}

void copy(void) {
  int i=0;
  extern char line[], longest[];
  while ((longest[i] = line[i]) != '\0')
    i++;
}
```

Struct Variable Definition : Sample

```
typedef struct{
  char *name;               // string label of GNode
  struct GEdge *edgelist; // pointer to list of GEdges
} GNode; // A single GNode with  label
struct GEdge{
  GNode *left;              // left-hand-GNode
  GNode *right;             // right hand-GNode
};              // A  GEdge in a directed graph
typedef struct GEdge GEdge;
typedef struct{
  char *name;    // name of graph
  GNode *nodes;   // array of GNodes
  GEdge *edges;   // array of GEdges
} Graph; // as name implies...
```

Q & A

```
int *f(void){
  int i;
  …..
  return &i;
}

void f(const  int *p){
    *p=0;
}
```

Q & A

Address T[I][J] = MatrixAccess(T, I, J) = T + (I * RS) + (J*M)

Let T be the 2 by 4 matrix. Assure the size of an Integer is 2 and the matrix is stored at location 1000 in memory.

String address : 1000

Number of rows : 2

Number of columns: 4

Type size : 2= sizeof(int)

Row size: 8= 2*4 //size of entire row

The addresses for the rows in storage are

Row 0:

Row 1:

The address of T[1][3] is

Review

- C versus Java
- C data types
- Functions
- Control Flow
- Scope
- Pointers, Arrays
- Strings