COS 217: Introduction to Programming Systems

Crash Course in C (Part 1)

The Design of C Language Features and
Data Types and their Operations and Representations

PRINCETON UNIVERSITY

1

Goals of this Lecture

Help you learn about:
- The decisions that were made by the designers* of C
- … and thereby…
- The fundamentals of C

Why?
- Learning the design rationale of the C language provides a richer understanding of C itself
- A mature programmer knows the philosophy of a language, not just the syntax

2 * Dennis Ritchie & subsequent members of standardization committees

2

The Design of C

"C is quirky, flawed, and an enormous success. While accidents of history surely helped, it evidently satisfied a need for a system implementation language efficient enough to displace assembly language, yet sufficiently abstract and fluent to describe algorithms and interactions in a wide variety of environments."

– Dennis Ritchie

Designers wanted C to:	But also:
Support system programming	Support application programming
Be low-level	Be portable
Be easy for people to handle	Be easy for computers to handle

3

3

DECLARATIONS AND ASSIGNMENTS

4

4

Declaring Variables

Decision: Should C require variable declarations? (Not all languages do!)

```
cmoretti@tars:~$awk 'END {print "\""myVariable"\""; print (myOtherVariable+0)}' < /dev/null
""
0
cmoretti@tars:~$echo \"$someNewVariable\"
""
```

Thought process:
- Declaring variables allows compiler to check "spelling"
- Declaring variables allows compiler to allocate memory more efficiently
- Declaring variables' types produces fewer surprises at runtime
- Declaring variables requires more from the programmer
  - Extra verbiage
  - Type foresight
  - "Do what I mean, not what I say"

5

5

Declaring Variables

Decisions:
- Require variable declarations
- Provide declaration statement
- Programmer specifies type of variable (and other attributes too)

Examples
- int i;
- int i, j;
- int i = 5;
- const int i = 5; /* value of i cannot change */
- static int i; /* covered later in course */
- extern int i; /* covered later in course */

6

6

Declaring Variables

Another Decision:
- Unlike Java, declaration statements in C90 must appear before any other kind of statement in compound statement

```
{
   int i;
   /* Non-declaration
      stmts that use i. */
   …
   int j;
   /* Non-declaration
      stmts that use j. */
   …
}
```

Illegal in C

```
{
   int i;
   int j;
   /* Non-declaration
      stmts that use i. */
   …
   /* Non-declaration
      stmts that use j. */
   …
}
```

Legal in C

7

7

Assignment

Issue: What about assignment?

Thought process
- Must have a way to assign a value to a variable
- Many high-level languages provide an assignment statement
- Would be more expressive to define an assignment *operator*
  - Performs assignment, and then evaluates to a value
  - Allows assignment to appear within larger expressions

Decisions
- Provide assignment operator
  - =
  - Variable on left, expression on right
- Define assignment operator to change the value of a variable, and emit the new value of that variable
- Right-to-left associativity

8

8

Assignment Operator Examples

Examples

```
i = 0;
   /* Side effect: assign 0 to i.
      Evaluate to 0.

j = i = 0;
   /* Side effect: assign 0 to i.
      Evaluate to 0.
      Side effect: assign 0 to j.
      Evaluate to 0. */

while ((i = getchar()) != EOF) …
   /* Read a character (maybe).
      Side effect: assign that character to i.
      Evaluate to that character.
      Compare that emitted value to EOF.
      Evaluate to 0 (FALSE) or 1 (TRUE). */
```

9

9

CONTROL STATEMENTS

10

10

Control Statements: History

What the computer does "under the hood":

```
/* add up numbers from
   1 to value in R2 */
1  R0 = 0
2  R1 = 1
3  compare R1, R2
4  if greater goto 8
5  R0 = R0 + R1
6  R1 = R1 + 1
7  goto 3
8  /* answer in R0 */
```

Early programming languages (1950's):

```
/* add up numbers from 1 to n */

sum = 0
i = 1
LOOP:
if (i > n) goto DONE
sum = sum + i
i = i + 1
goto LOOP
DONE:
/* answer in sum */
```

Some high-level conveniences (variable names, labels) but control flow based on `if` and `goto`

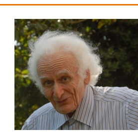
11

11

Control Statements

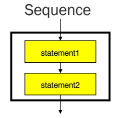
Algol-60 language (1960)
- BEGIN-END, IF-THEN-ELSE, WHILE-DO, FOR, (and also GOTO)

Scientific background
- Böhm and Jacopini proved (1966) that any algorithm can be expressed as the nesting of only 3 control structures:

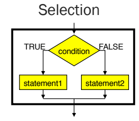


Corrado Böhm

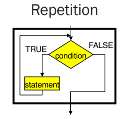
Sequence



Selection



Repetition

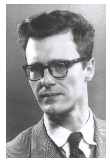


12

12

Control Statements (cont.)

Thought Process
- Dijkstra argued that any algorithm should be expressed using **only** those control structures (Go To Statement Considered Harmful, 1968)


Edsger Dijkstra

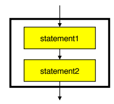
C language design (1972)
- Basically follow ALGOL-60, but use { braces } instead of the more heavyweight BEGIN – END

13

13

Sequence Statement

Compound statement, alias block



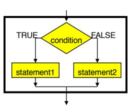

```
{
   statement1;
   statement2;
   …
}
```

14

14

Selection Statements

`if` and `if-else` statements




```
if (expr)
   statement1;
```

```
if (expr)
   statement1;
else
   statement2;
```

15

15

Selection Statements

`switch` and `break` statements
- for multi-path decisions on a single integer expression

```
switch (integerExpr)
{  case integerLiteral1:
      …
      break;
   case integerLiteral2:
      …
      break;
   …
   default:
      …
}
```

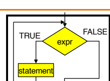
What happens if you forget to **`break`**?

16

16

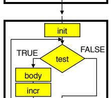
Repetition Statements

- while statement: test at leading edge



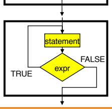

```
while (expr)
   statement;
```

- for statement: test at leading edge, increment at trailing




```
for (initExpr; testExpr; incrExpr)
   bodyStatement;
```

- do-while statement: test at trailing edge




```
do
   statement;
while (expr);
```

17

17

Repetition Statements

Cascading implications
- Declarations must come at the beginning of a block → cannot declare loop control variable in for statement

```
{
   …
   for (int i = 0; i < 10; i++)
      /* Do something */
   …
}
```

Illegal in C

```
{
   int i;
   …
   for (i = 0; i < 10; i++)
      /* Do something */
   …
}
```

Legal in C

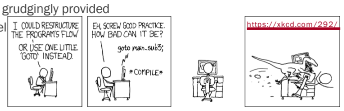
18

18

Other Control Statements

Issue: What other control statements should C provide?

Decisions
- break statement
  - Breaks out of closest enclosing switch or repetition statement
- continue statement
  - Goes back to condition check, skipping remainder of current iteration
  - When used within for, still executes increment step
- goto statement grudgingly provided
  - Jump to label



19

19

I/O

20

20

I/O Facilities

Decisions
- Do not provide I/O facilities in the language
- Instead provide I/O facilities in standard library
  - Constant: EOF
  - Data type: FILE (described later in course)
  - Variables: stdin, stdout, and stderr
  - Functions: …

21

21

Reading Data Types

Issue: What functions should C provide for reading data of primitive types?

Thought process
- Must convert external form (sequence of character codes) to internal form
- Could provide getchar(), getshort(), getint(), getfloat(), etc.
- Could provide one parameterized function to read any primitive type of data

Decisions
- Provide scanf() function
- Can read any primitive type of data
- First parameter is a format string containing conversion specs

See King book for details

22

22

Reading Characters

Issue: Should reading characters be granted special status?

Thought process
- Desirable to have a function to read a single byte from stdin
- Function must have a way to indicate failure, that is, to indicate that no bytes remain

Decisions
- Provide getchar() function
- Make return type of getchar() wider than char
  - Make it int; that's the natural word size
- Define getchar() to return EOF (a special non-character int) to indicate failure

Reminder: there is no such thing as "the EOF character"

23

23

Writing Data Types

Issue: What functions should C provide for writing data of primitive types?

Thought process
- Must convert internal form to external form (sequence of character codes)
- Could provide putchar(), putshort(), putint(), putfloat(), etc.
- Could provide one parameterized function to write any primitive type of data

Decisions
- Provide printf() function
- Can write any primitive type of data
- First parameter is a format string containing conversion specs

See King book for details

24

24

Writing Characters

Issue: What functions should C provide for writing a character to standard output?

Thought process
- Desirable to have a function to write a single character to stdout

Decisions
- Provide a putchar() function
- Define putchar() to accept one parameter
  - For symmetry with getchar(), parameter is an int

25

25

Other I/O Facilities

Issue: What other I/O functions should C provide?

Decisions
- fopen(): Open a stream
- fclose(): Close a stream
- fgetc(): Read a character from specified stream
- fputc(): Write a character to specified stream
- fgets(): Read a line/string from specified stream
- fputs(): Write a line/string to specified stream
- fscanf(): Read data from specified stream
- fprintf(): Write data to specified stream

Described in King book, and later in the course
after covering files, arrays, and strings

26

26

Statements Summary: C vs. Java

Java only
- Declarations anywhere within block
- Declare immutable variables with final
- Conditionals of type boolean
- "Labeled" break and continue
- No goto

C only
- Declarations only at beginning block
- Declare immutable variables with const
- Conditionals of any type (checked for zero / nonzero)
- No "labeled" break and continue
- goto provided (but don't use it except in flattened C at end of course)

27

27

Q: Why do computer programmers confuse Christmas and Halloween?

A: Because 25 Dec == 31 Oct

NUMBER SYSTEMS

28

28

The Decimal Number System

Name
- "decem" (Latin) ⇒ ten

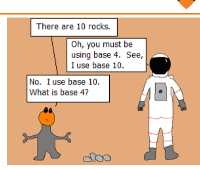
Characteristics
- For us, these symbols (Not universal ...)
  - `0 1 2 3 4 5 6 7 8 9`

https://bit.ly/3ifUw1b

European (descended from the West Arabic)	0	1	2	3	4	5	6	7	8	9
Arabic-Indic	٠	١	٢	٣	٤	٥	٦	٧	٨	٩
Eastern Arabic-Indic (Persian and Urdu)	۰	۱	۲	۳	۴	۵	۶	۷	۸	۹
Devanagari (Hindi)	०	१	२	३	४	५	६	७	८	९
Tamil		௧	௨	௩	௪	௫	௬	௭	௮	௯

- Positional
  - `2945 ≠ 2495`
  - `2945 = (2*10³) + (9*10²) + (4*10¹) + (5*10⁰)`

(Most) people use the decimal number system



Every base is base 10.

29

29

The Binary Number System

binary

adjective: being in a state of one of two mutually exclusive conditions such as on or off, true or false, molten or frozen, presence or absence of a signal.
From Late Latin *bīnārius* ("consisting of two").

Characteristics
- Two symbols: `0 1`
- Positional: $1010_B \neq 1100_B$

Most (digital) computers use the binary number system

Terminology
- **Bit**: a single binary symbol ("binary digit")
- **Byte**: (typically) 8 bits
- **Nybble**: 4 bits

Why?

30

30

Decimal-Binary Equivalence

Decimal	Binary
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

Decimal	Binary
16	10000
17	10001
18	10010
19	10011
20	10100
21	10101
22	10110
23	10111
24	11000
25	11001
26	11010
27	11011
28	11100
29	11101
30	11110
31	11111
...	...

31

31

Decimal-Binary Conversion

Binary to decimal: expand using positional notation

$$100101_B = (1*2^5)+(0*2^4)+(0*2^3)+(1*2^2)+(0*2^1)+(1*2^0)$$
$$= 32 + 0 + 0 + 4 + 0 + 1$$
$$= 37$$

Most-significant bit (msb)

Least-significant bit (lsb)

32

32

Integer-Binary Conversion

(Decimal) Integer to binary: do the reverse

- Determine largest power of 2 that's ≤ number; write template

$$37 = (?*2^5)+(?*2^4)+(?*2^3)+(?*2^2)+(?*2^1)+(?*2^0)$$

- Fill in template

$$37 = (1*2^5)+(0*2^4)+(0*2^3)+(1*2^2)+(0*2^1)+(1*2^0)$$

```
 -32
   5
  -4
   1
  -1
   0
```

100101_B

34

34

Integer-Binary Conversion

Integer to binary division method

- Repeatedly divide by 2, consider remainder

```
37 / 2 = 18 R 1
18 / 2 =  9 R 0
 9 / 2 =  4 R 1
 4 / 2 =  2 R 0
 2 / 2 =  1 R 0
 1 / 2 =  0 R 1
```

Read from bottom to top: 100101_B

35

35

The Hexadecimal Number System

Name

- "hexa-" (Ancient Greek ἑξα-) ⇒ six
- "decem" (Latin) ⇒ ten

Characteristics

- Sixteen symbols
  - 0 1 2 3 4 5 6 7 8 9 A B C D E F
- Positional
  - A13DH ≠ 3DA1H

Computer programmers often use hexadecimal or "hex"

- In C: 0x prefix (0xA13D, etc.)

Why?

36

36

Decimal-Hexadecimal Equivalence

Decimal	Hex
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F

Decimal	Hex
16	10
17	11
18	12
19	13
20	14
21	15
22	16
23	17
24	18
25	19
26	1A
27	1B
28	1C
29	1D
30	1E
31	1F

Decimal	Hex
32	20
33	21
34	22
35	23
36	24
37	25
38	26
39	27
40	28
41	29
42	2A
43	2B
44	2C
45	2D
46	2E
47	2F
...	...

37

37

Integer-Hexadecimal Conversion

Hexadecimal to (decimal) integer: expand using positional notation

```
25H = (2*16^1) + (5*16^0)
    =   32     +    5
    =   37
```

Integer to hexadecimal: use the division method

```
37 / 16 = 2 R 5
 2 / 16 = 0 R 2
```

Read from bottom to top: 25_H

38

38

Binary-Hexadecimal Conversion

Observation: 161 = 24
- Every 1 hexit corresponds to 4 bits

Binary to hexadecimal

```
1010000100111101B
  A   1   3   DH
```

Digit count in binary number not a multiple of 4 ⇒ pad with zeros on left

Hexadecimal to binary

```
  A   1   3   DH
1010000100111101B
```

Discard leading zeros from binary number if appropriate

39

39

Base Conversion Quick Quiz

Convert binary 101010 into decimal and hex

A. 21 decimal, 1A hex
B. 42 decimal, 2A hex
C. 48 decimal, 32 hex
D. 55 decimal, 4G hex

hint: convert to hex first

10 1010
2 A
32 + 10 = 42

2 + 8 + 32 = 42

40

40

The Octal Number System

Name
- "octo" (Latin) ⇒ eight

Characteristics
- Eight symbols
  - 0 1 2 3 4 5 6 7
- Positional
  - 17430 ≠ 73140

Computer programmers often use octal (so does Mickey!)
- In C: 0 prefix (01743, etc.)

Why?

```
cmoretti@tars:tmp$ls -l myFile
-rw-r--r--  1 cmoretti  wheel  0 Sep  7 10:58 myFile
cmoretti@tars:tmp$chmod 755 myFile
cmoretti@tars:tmp$ls -l myFile
-rwxr-xr-x  1 cmoretti  wheel  0 Sep  7 10:58 myFile
```

41

41