CENG 230

Introduction to C Programming

Week 1 – Overview and introduction

Sinan Kalkan

Some slides/content are borrowed from Tansel Dokeroglu, Nihan Kesim Cicekli.
Syllabus
How to study?

• Follow the lectures and the labs
• Read the textbook on a weekly basis
• Get your hands dirty
  • Do the exercises in front of the computer
Appointment

• No office hours. Make an appointment.
• Via email: skalkan@ceng.metu.edu.tr
• Office:
  Room B207,
  Department of Computer Engineering
• WWW:
  • http://kovan.ceng.metu.edu.tr/~sinan/
Programming, computation, algorithm
Program, Programming

```
int alice = 1;
int bob = 456;
int carol;
main(void)
{
    carol = alice*bob;
    printf("%d", carol);
}
```
What is an algorithm?

- An algorithm is a list that looks like
  - STEP 1: Do something
  - STEP 2: Do something
  - STEP 3: Do something
  - .
  - .
  - .
  - .
  - STEP N: Stop, you are finished

From “Invitation to Computer Science”
"I think you should be more explicit here in step two.

Then a miracle occurs.

\[
\begin{align*}
203 & - 624 \\
2X & = 56 \\
\sqrt{0.6511} & \approx 0.8017 \\
\frac{345}{11} & = 31.36
\end{align*}
\]
A formal definition of algorithm

• “Starting from an initial state and initial input (perhaps empty), the instructions describe a computation that, when executed, will proceed through a finite number of well-defined successive states, eventually producing "output" and terminating at a final ending state.”
Algorithms

• We use them all the time.
• Can you give examples?
  • Following directions
  • Recording a DVD
  • Adding two numbers
  • Finding Greatest Common Divisor
  • ...

From “Invitation to Computer Science”
An example algorithm

Algorithm for Adding Two $m$-Digit Numbers

Given: $m \geq 1$ and two positive numbers each containing $m$ digits, $a_{m-1} a_{m-2} \ldots a_0$ and $b_{m-1} b_{m-2} \ldots b_0$

Wanted: $c_m c_{m-1} c_{m-2} \ldots c_0$, where $c_m c_{m-1} c_{m-2} \ldots c_0 = (a_{m-1} a_{m-2} \ldots a_0) + (b_{m-1} b_{m-2} \ldots b_0)$

Algorithm:

Step 1 Set the value of carry to 0.
Step 2 Set the value of $i$ to 0.
Step 3 While the value of $i$ is less than or equal to $m - 1$, repeat the instructions in steps 4 through 6.
Step 4 Add the two digits $a_i$ and $b_i$ to the current value of carry to get $c_i$.
Step 5 If $c_i \geq 10$, then reset $c_i$ to $(c_i - 10)$ and reset the value of carry to 1; otherwise, set the new value of carry to 0.
Step 6 Add 1 to $i$, effectively moving one column to the left.
Step 7 Set $c_m$ to the value of carry.
Step 8 Print out the final answer, $c_m c_{m-1} c_{m-2} \ldots c_0$.
Step 9 Stop.

From “Invitation to Computer Science”
“Computation”

• Digital vs. analog computation
• Sequential vs. parallel computation
• Batch vs. interactive computation
• Evolutionary, molecular, quantum computation
• “Physical computation” / “Digital Physics”
  • ‘The whole universe is itself a computation’
Computation in our brain

• Highly-connected network of neurons.
• How many neurons?
  • Approx. $10^{11}$ neurons and $10^{14}$ synapses.
• How do they transmit information?
  • Using nothing else than charged molecules.
Computation in our brain (cont’d)

• Each neuron gets input and produces an output using an “activation function”
Turing Machine

Von Neumann Architecture

DIGITAL COMPUTATION
But first some historical overview
The Early Period: Up to 1940

• 3,000 years ago: Mathematics, logic, and numerical computation
  • Important contributions made by the Greeks, Egyptians, Babylonians, Indians, Chinese, and Persians
  • Cuneiform
  • Stone “abacus”

• [http://www.thocp.net/slideshow/0469.htm](http://www.thocp.net/slideshow/0469.htm)
ABACUS

Early calculating devices
ABACUS – 2700 BC (Mesopotamia)

http://www.computersciencelab.com/ComputerHistory/History.htm

Slide from “Introduction to Computing”
DaVinci

• 1452-1519 Leonardo DaVinci sketched gear-driven calculating machines but none were ever built.

http://www.computersciencelab.com/ComputerHistory/History.htm
Napier’s Bones

• 1614: Logarithms
  • Invented by John Napier to simplify difficult mathematical computations

http://www.computersciencelab.com/ComputerHistory/History.htm

Slide from “Introduction to Computing”
• If you want to multiply 7 by 46785499:
Slide Rule (slipstick) “a mechanical analog computer”

Around 1622: First slide rule created

http://www.computersciencelab.com/ComputerHistory/History.htm
The Pascaline: One of the Earliest Mechanical Calculators
The Early Period: Up to 1940

Jacquard’s Loom

Also see http://www.computersciencelab.com/ComputerHistory/HistoryPt2.htm
Difference engine

http://www.youtube.com/watch?v=0anIyVGeWOI
The Harvard Mark-I

Grace M. Hopper working on the Harvard Mark-I, developed by IBM and Howard Aiken. The Mark-I remained in use at Harvard until 1959, even though other machines had surpassed it in performance, providing vital calculations for the navy in World War II.
Programming the ENIAC
DIGITAL COMPUTATION
A computer

Devices

Gates

Transistors
Everything in a PC is Binary ... well, almost ...

<table>
<thead>
<tr>
<th>States of a Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>
A transistor

This circuit functions as a switch. In other words, based on the control voltage, the circuit either passes Vin to output or not.
Examples of transistors

Replica of the first transistor

A set of transistors, depicting the fast change in technology.
A computer

Devices

Gates

Transistors
NOT Gate

<table>
<thead>
<tr>
<th>X</th>
<th>(\bar{X})</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

\[ x \quad \rightarrow \quad \bar{x} \]
AND gate

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>X·Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
OR Gate

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>X+Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

![OR Gate Diagram]
### XOR Gate

The XOR (exclusive OR) gate is a digital logic gate that outputs true (1) only when the inputs differ. The truth table for an XOR gate is as follows:

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>X ⊕ Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

[Diagram of XOR gate]
An example problem: Water Tank

<table>
<thead>
<tr>
<th>HI</th>
<th>LO</th>
<th>Pump</th>
<th>Drain</th>
<th>Truth Table Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Tank level is OK</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Low level, pump more in</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>High level, drain some out</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>x</td>
<td>x</td>
<td>Inputs cannot occur</td>
</tr>
</tbody>
</table>

Schematic Representation

CENG 230 - Spring 2015
The binary addition

\[
\begin{array}{cccc}
0 & 1 & 0 & 1 \\
+0 & +0 & +1 & +1 \\
\hline
0 & 1 & 1 & 10 \\
\end{array}
\]

Question (Binary notation) : \(111010 + 11011 = ?\)
1-bit full-adder

A | B | CI | S | CO
---|---|----|---|---
0  | 0 | 0  | 0 | 0
0  | 0 | 1  | 1 | 0
0  | 1 | 0  | 1 | 0
0  | 1 | 1  | 0 | 1
1  | 0 | 0  | 1 | 0
1  | 0 | 1  | 0 | 1
1  | 1 | 0  | 0 | 1
1  | 1 | 1  | 1 | 1
N-bit Adder
Representing data
Data Representation

• Based on 1s and 0s
  • So, everything is represented as a set of binary numbers

• We will now see how we can represent:
  • Integers: 3, 1234435, -12945 etc.
  • Floating point numbers: 4.5, 124.3458, -1334.234 etc.
  • Characters: /, &+, -, A, a, ^, 1, etc.
  • ...

Binary Representation of Numeric Information

• Decimal numbering system
  • Base-10
  • Each position is a power of 10
    \[3052 = 3 \times 10^3 + 0 \times 10^2 + 5 \times 10^1 + 2 \times 10^0\]

• Binary numbering system
  • Base-2
  • Uses ones and zeros
  • Each position is a power of 2
    \[1101 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0\]
Decimal-to-binary Conversion

• Divide the number until zero:
  • $35 / 2 = 17 \times 2 + 1$
  • $17 / 2 = 8 \times 2 + 1$
  • $8 / 2 = 4 \times 2 + 0$
  • $4 / 2 = 2 \times 2 + 0$
  • $2 / 2 = 1 \times 2 + 0$

• Therefore, 35 has the binary representation: $100011$
IEEE 32bit Floating-Point Number Representation

- **Sign** (8 bits)
- **Exponent (E)**: 8 bits
  - Add 127 to the exponent value before storing it
  - E can be 0 to 255 with 127 representing the real zero.
- **Fraction (M - Mantissa)**: 23 bits

\[
(-1)^{\text{sign}}(1.b_{-1}b_{-2}...b_{-23})_2 \times 2^{e-127}
\]

- \(M \times 2^E\)
- Exponent (E): 8 bits
  - Add 127 to the exponent value before storing it
  - E can be 0 to 255 with 127 representing the real zero.
- Fraction (M - Mantissa): 23 bits
- \(2^{128} = 1.70141183 \times 10^{38}\)

Example:

- Binary: 0 0 1 1 1 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
- Decimal: 0.15625

\[
= 0.15625 = \frac{\text{Decimal}}{2^{128}}
\]
IEEE 32bit Floating-Point Number Representation

- Example: 12.375
- The digits before the dot:
  - \((12)_{10} \rightarrow (1100)_{2}\)
- The digits after the dot:
  - 1\textsuperscript{st} Way: 0.375 \rightarrow 0 \times \frac{1}{2} + 1 \times \frac{1}{4} + 1 \times \frac{1}{8} \rightarrow 011
  - 2\textsuperscript{nd} Way: Multiply by 2 and get the integer part until 0:
    - 0.375 \times 2 = 0.750 = 0 + 0.750
    - 0.750 \times 2 = 1.50 = 1 + 0.50
    - 0.50 \times 2 = 1.0 = 1 + 0.0
- \((12.375)_{10} = (1100.011)_{2}\)
- NORMALIZE: \((1100.011)_{2} = (1.100011)_{2} \times 2^{3}\)
- Exponent: 3, adding 127 to it, we get 1000 0010
- Fraction: 100011
- Then our number is: 0 10000010 100011000000000000000000
Binary Representation of Textual Information

• Characters are mapped onto binary numbers
  • ASCII (American Standard Code for Information Interchange) code set
    • Originally: 7 bits per character; 128 character codes
  • Unicode code set
    • 16 bits per character
  • UTF-8 (UCS Transformation Format) code set.
    • Variable number of 8-bits.
### Binary Representation of Textual Information (cont’d)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>48</td>
<td>00110000</td>
<td>0</td>
<td>0x30</td>
<td>0x0030</td>
<td>0</td>
</tr>
<tr>
<td>49</td>
<td>00110001</td>
<td>1</td>
<td>0x31</td>
<td>0x0031</td>
<td>1</td>
</tr>
<tr>
<td>50</td>
<td>00110010</td>
<td>2</td>
<td>0x32</td>
<td>0x0032</td>
<td>2</td>
</tr>
<tr>
<td>51</td>
<td>00110011</td>
<td>3</td>
<td>0x33</td>
<td>0x0033</td>
<td>3</td>
</tr>
<tr>
<td>52</td>
<td>00110100</td>
<td>4</td>
<td>0x34</td>
<td>0x0034</td>
<td>4</td>
</tr>
<tr>
<td>53</td>
<td>00110101</td>
<td>5</td>
<td>0x35</td>
<td>0x0035</td>
<td>5</td>
</tr>
<tr>
<td>54</td>
<td>00110110</td>
<td>6</td>
<td>0x36</td>
<td>0x0036</td>
<td>6</td>
</tr>
<tr>
<td>55</td>
<td>00110111</td>
<td>7</td>
<td>0x37</td>
<td>0x0037</td>
<td>7</td>
</tr>
<tr>
<td>56</td>
<td>00111000</td>
<td>8</td>
<td>0x38</td>
<td>0x0038</td>
<td>8</td>
</tr>
<tr>
<td>57</td>
<td>00111001</td>
<td>9</td>
<td>0x39</td>
<td>0x0039</td>
<td>9</td>
</tr>
<tr>
<td>58</td>
<td>00111010</td>
<td>:</td>
<td>0x3A</td>
<td>0x003A</td>
<td>:</td>
</tr>
<tr>
<td>59</td>
<td>00111011</td>
<td>;</td>
<td>0x3B</td>
<td>0x003B</td>
<td>;</td>
</tr>
<tr>
<td>60</td>
<td>00111100</td>
<td>&lt;</td>
<td>0x3C</td>
<td>0x003C</td>
<td>&lt;</td>
</tr>
<tr>
<td>61</td>
<td>00111101</td>
<td>=</td>
<td>0x3D</td>
<td>0x003D</td>
<td>=</td>
</tr>
<tr>
<td>62</td>
<td>00111110</td>
<td>&gt;</td>
<td>0x3E</td>
<td>0x003E</td>
<td>&gt;</td>
</tr>
<tr>
<td>63</td>
<td>00111111</td>
<td>?</td>
<td>0x3F</td>
<td>0x003F</td>
<td>?</td>
</tr>
<tr>
<td>64</td>
<td>01000000</td>
<td>@</td>
<td>0x40</td>
<td>0x0040</td>
<td>@</td>
</tr>
<tr>
<td>65</td>
<td>01000001</td>
<td>A</td>
<td>0x41</td>
<td>0x0041</td>
<td>A</td>
</tr>
<tr>
<td>66</td>
<td>01000010</td>
<td>B</td>
<td>0x42</td>
<td>0x0042</td>
<td>B</td>
</tr>
</tbody>
</table>

*Partial listings only!*

**ASCII** 7 bits long

**Unicode** 16 bits long
## Integer Types

Following table gives you details about standard integer types with its storage sizes and value ranges:

<table>
<thead>
<tr>
<th>Type</th>
<th>Storage size</th>
<th>Value range</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>1 byte</td>
<td>-128 to 127 or 0 to 255</td>
</tr>
<tr>
<td>unsigned char</td>
<td>1 byte</td>
<td>0 to 255</td>
</tr>
<tr>
<td>signed char</td>
<td>1 byte</td>
<td>-128 to 127</td>
</tr>
<tr>
<td>int</td>
<td>2 or 4 bytes</td>
<td>-32,768 to 32,767 or -2,147,483,648 to 2,147,483,647</td>
</tr>
<tr>
<td>unsigned int</td>
<td>2 or 4 bytes</td>
<td>0 to 65,535 or 0 to 4,294,967,295</td>
</tr>
<tr>
<td>short</td>
<td>2 bytes</td>
<td>-32,768 to 32,767</td>
</tr>
<tr>
<td>unsigned short</td>
<td>2 bytes</td>
<td>0 to 65,535</td>
</tr>
<tr>
<td>long</td>
<td>4 bytes</td>
<td>-2,147,483,648 to 2,147,483,647</td>
</tr>
<tr>
<td>unsigned long</td>
<td>4 bytes</td>
<td>0 to 4,294,967,295</td>
</tr>
</tbody>
</table>

## Floating-Point Types

Following table gives you details about standard floating-point types with storage sizes and value ranges and their precision:

<table>
<thead>
<tr>
<th>Type</th>
<th>Storage size</th>
<th>Value range</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td>4 byte</td>
<td>1.2E-38 to 3.4E+38</td>
<td>6 decimal places</td>
</tr>
<tr>
<td>double</td>
<td>8 byte</td>
<td>2.3E-308 to 1.7E+308</td>
<td>15 decimal places</td>
</tr>
<tr>
<td>long double</td>
<td>10 byte</td>
<td>3.4E-4932 to 1.1E+4932</td>
<td>19 decimal places</td>
</tr>
</tbody>
</table>
Computer organization
Logical organization of computer
Memory and Cache (continued)

• RAM (Random Access Memory)
  Often called *memory, primary memory*
  
  • Memory made of addressable “cells”
  
  • Cell size is 8 bits
    • Nowadays, it is 32 or 64 bits.
  
  • All memory cells accessed in equal time
  
  • Memory address
    • Unsigned binary number N long
    • Address space is then $2^N$ cells
Programming
Program, Programming

```
int alice = 1;
int bob = 456;
int carol;
main(void)
{
  carol = alice*bob;
  printf("%d", carol);
}
```
Machine language  | Assembly language  | Programming languages such as C++, Java | Pseudocode | English, Spanish, Japanese, ...  

Low-level languages (closely related to the hardware) | High-level languages (more removed from details of the hardware) | Natural languages (not related to the hardware)
main:
    pushq    %rbp
    movq     %rsp, %rbp
    movl     alice(%rip), %edx
    movl     bob(%rip), %eax
    imull    %edx, %eax
    movl     %eax, carol(%rip)
    movl     $0, %eax
    leave
    ret

alice:
    .long    123

bob:
    .long    456

int alice = 123;
int bob = 456;
int carol;
main(void)
{
    carol = alice*bob;
}

...
How are languages implemented

COMPILATIVE APPROACH

```c
int alice = 123;
int bob = 456;
int carol;
main(void)
{
    carol = alice*bob;
    printf("%d", carol);
}
```
C language development environment
The Translation/Loading/Execution Process
## Algorithmic notation

<table>
<thead>
<tr>
<th>Address</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Value of $a$</td>
</tr>
<tr>
<td>101</td>
<td>Value of $b$</td>
</tr>
<tr>
<td>102</td>
<td>Value of $c$</td>
</tr>
</tbody>
</table>

## Machine Language Instruction Sequences

<table>
<thead>
<tr>
<th>Address</th>
<th>Contents</th>
<th>Commentary</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>LOAD 101</td>
<td>Put the value of $b$ into register R.</td>
</tr>
<tr>
<td>51</td>
<td>ADD 102</td>
<td>Add $c$ to register R. It now holds $b + c$.</td>
</tr>
<tr>
<td>52</td>
<td>STORE 100</td>
<td>Store the contents of register R into $a$.</td>
</tr>
</tbody>
</table>

1. Set $a$ to the value $b + c$

2. If $a > b$ then
   - set $c$ to the value $a$
   - Else
     - set $c$ to the value $b$

   - Go to location 54 if $a > b$.
   - Get here if $a \leq b$, so move $b$ into $c$ and skip the next instruction.
   - Move $a$ into $c$. Next statement begins here.
Bugs, Errors

• Syntax Errors
  Area = 3.1415 * R * R
  Area = 3.1415 x R x R

• Run-time Errors

```python
>>> def SqrtDelta(a,b,c):
    return sqrt(b*b - 4*a*c)
>>> print SqrtDelta(1,3,1)
2.2360679774997898
>>> print SqrtDelta(1,1,1)
ValueError: math domain error
```
Bugs, Errors

• Logical Errors

$$\text{root}_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

```python
>>> root1 = (-b + sqrt(b*b - 4*a*c)) / 2*a
```

• Design Errors

$$x^3 + ax^2 + bx + c = 0$$

$$\text{root}_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$
History of C

• C
  – Developed by Denis M. Ritchie at AT&T Bell Labs in 1972 as a systems programming language
  – Used to develop UNIX
  – Used to write modern operating systems
  – Hardware independent (portable)

• Standardization
  – Many slight variations of C existed, and were incompatible
  – Committee formed to create a "unambiguous, machine independent“ definition
  – Standard created in 1989, updated in 1999
/* Fig. 2.1: fig02_01.c
   A first program in C */
#include <stdio.h>

/* function main begins program execution */
int main( void )
{
    printf( "Welcome to C!\n" );

    return 0; /* indicate that program ended successfully */
} /* end function main */

Welcome to C!

Fig. 2.1  |  A first program in C.