COMSW 1003-1

Introduction to Computer Programming in C

Lecture 3

Spring 2011

Instructor: Michele Merler

http://www1.cs.columbia.edu/~mmerler/comsw1003-1.html
Today

• Computer Architecture (Brief Overview)
• “Hello World” in detail
• C Syntax
• Variables and Types
• Operators
• printf (if there is time)
Von Neumann Architecture

- Data
- Instructions
Computer Memory Architecture

- CPU Registers
- Level 1 Cache
- Level 2 Cache
- Physical RAM (Main Memory)
- Disc Storage (Virtual RAM, Hard Drive)

Cost per byte, access speed vs. Capacity
Von Neumann Architecture

The Program Counter (PC) **points** (= tells the CPU) to the address in memory where the next instruction to be executed resides.
Hello World

#include <stdio.h>

int main()
{
    printf("Hello World\n");
    return(0);
}
Hello World

#include <stdio.h>

int main(){
    printf("Hello World\n");
    return(0);
}
The Operating System

- Manages the hardware
- Allocates resources to programs
- Accommodates user requests
- First program to be executed when computer starts
  (loaded from ROM)

- Windows
- Unix
- Mac OS
- Android
- Linux
- Solaris
- Chrome OS

Diagram:
- User
- Operating System (OS)
- Hardware
- Program
Hello World

Global Definitions

- \#include <stdio.h>

Body of function

- `int main(){`
  - `printf("Hello World\n");`
  - `return(0);`

External Header
- (standard C library containing functions for Input/Output)

Function definition:
- • It’s called `main`
- • It does not take any input ()
- • It returns an integer

Single statements
C Syntax

• Statements
  – one line commands
  – always end with ;
  – can be grouped between { }
  – spaces are not considered

• Comments
  // single line comment

  /* multiple lines comments */
Hello World + Comments

/*
 *  My first C program
 * /

#include <stdio.h>

int main(){

    printf("Hello World\n");

    return(0);  // return 0 to the OS = OK

}
Variables and types

• **Variables** are placeholders for values
  
  ```c
  int x = 2;
  x = x + 3; // x value is 5 now
  ```

• In C, variables are divided into **types**, according to how they are **represented in memory** (always represented in binary)
  
  – int
  – float
  – double
  – char
Variables Declaration

• Before we can use a variable, we must declare (= create) it

• When we declare a variable, we specify its type and its name

```c
int x;
float y = 3.2;
```

• Most of the time, the compiler also allocates memory for the variable when it’s declared. In that case declaration = definition

• There exist special cases in which a variable is declared but not defined, and the computer allocates memory for it only at run time (will see with functions and external variables)
int

• No fractional part or decimal point (ex. +3, -100)
• Represented with 4 bytes (32 bits) in UNIX

• Sign
  – **unsigned**: represents only positive values, all bits for value
    Range: from 0 to 2^{32}
  – **signed** (default): 1 bit for sign + 31 for actual value
    Range: from -2^{31} to 2^{31}

• Size
  – **short** int: at least 16 bits
  – **long** int: at least 32 bits
  – **long long** int: at least 64 bits
  – size(short) ≤ size(int) ≤ size(long)

\[
\begin{align*}
\text{int } & x = -12; \\
\text{unsigned int } & x = 5; \\
\text{short (int) } & x = 2;
\end{align*}
\]
**float**

- Single precision floating point value
- Fractional numbers with decimal point
- Represented with 4 bytes (32 bits)
- Range: \(-10^{38}\) to \(10^{38}\)
- Exponential notation: \(-0.278 \times 10^3\)

\[
\frac{f}{m} = 11.5;
\]

\[
n_{10} = (-1)^s \cdot (f \cdot 2^{-23}) \cdot 2^{m-127}
\]
Double

- Double precision floating point
- Represented with 8 bytes (64 bits)

```c
double x = 121.45;
```
char

- Character
- Single byte representation
- 0 to 255 values expressed in the ASCII table

```c
char c = 'w' ;
```
## ASCII Table

<table>
<thead>
<tr>
<th>Dec</th>
<th>Hex</th>
<th>Oct</th>
<th>Char</th>
<th>Dec</th>
<th>Hex</th>
<th>Oct</th>
<th>Html</th>
<th>Chr</th>
<th>Dec</th>
<th>Hex</th>
<th>Oct</th>
<th>Html</th>
<th>Chr</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
<td>000</td>
<td>NULL (null)</td>
<td>32</td>
<td>20</td>
<td>040</td>
<td> </td>
<td>Space</td>
<td>64</td>
<td>40</td>
<td>100</td>
<td> </td>
<td>@</td>
</tr>
<tr>
<td>1</td>
<td>01</td>
<td>001</td>
<td>SOH (start of heading)</td>
<td>33</td>
<td>21</td>
<td>041</td>
<td> </td>
<td>!</td>
<td>65</td>
<td>41</td>
<td>101</td>
<td> </td>
<td>A</td>
</tr>
<tr>
<td>2</td>
<td>02</td>
<td>002</td>
<td>STX (start of text)</td>
<td>34</td>
<td>22</td>
<td>042</td>
<td> </td>
<td>&quot;</td>
<td>66</td>
<td>42</td>
<td>102</td>
<td> </td>
<td>B</td>
</tr>
<tr>
<td>3</td>
<td>03</td>
<td>003</td>
<td>ETX (end of text)</td>
<td>35</td>
<td>23</td>
<td>043</td>
<td> </td>
<td>#</td>
<td>67</td>
<td>43</td>
<td>103</td>
<td> </td>
<td>C</td>
</tr>
<tr>
<td>4</td>
<td>04</td>
<td>004</td>
<td>EOT (end of transmission)</td>
<td>36</td>
<td>24</td>
<td>044</td>
<td> </td>
<td>$</td>
<td>68</td>
<td>44</td>
<td>104</td>
<td> </td>
<td>D</td>
</tr>
<tr>
<td>5</td>
<td>05</td>
<td>005</td>
<td>ENQ (enquiry)</td>
<td>37</td>
<td>25</td>
<td>045</td>
<td> </td>
<td>%</td>
<td>69</td>
<td>45</td>
<td>105</td>
<td> </td>
<td>E</td>
</tr>
<tr>
<td>6</td>
<td>06</td>
<td>006</td>
<td>ACK (acknowledge)</td>
<td>38</td>
<td>26</td>
<td>046</td>
<td> </td>
<td>&amp;</td>
<td>70</td>
<td>46</td>
<td>106</td>
<td> </td>
<td>F</td>
</tr>
<tr>
<td>7</td>
<td>07</td>
<td>007</td>
<td>BEL (bell)</td>
<td>39</td>
<td>27</td>
<td>047</td>
<td> </td>
<td>'</td>
<td>71</td>
<td>47</td>
<td>107</td>
<td> </td>
<td>G</td>
</tr>
<tr>
<td>8</td>
<td>08</td>
<td>010</td>
<td>BS (backspace)</td>
<td>40</td>
<td>28</td>
<td>050</td>
<td> </td>
<td>(</td>
<td>72</td>
<td>48</td>
<td>110</td>
<td> </td>
<td>H</td>
</tr>
<tr>
<td>9</td>
<td>09</td>
<td>011</td>
<td>TAB (horizontal tab)</td>
<td>41</td>
<td>29</td>
<td>051</td>
<td> </td>
<td>)</td>
<td>73</td>
<td>49</td>
<td>111</td>
<td> </td>
<td>I</td>
</tr>
<tr>
<td>10</td>
<td>A</td>
<td>012</td>
<td>LF (NL line feed, new line)</td>
<td>42</td>
<td>2A</td>
<td>052</td>
<td> </td>
<td>*</td>
<td>74</td>
<td>4A</td>
<td>112</td>
<td> </td>
<td>J</td>
</tr>
<tr>
<td>11</td>
<td>B</td>
<td>013</td>
<td>VT (vertical tab)</td>
<td>43</td>
<td>2B</td>
<td>053</td>
<td> </td>
<td>+</td>
<td>75</td>
<td>4B</td>
<td>113</td>
<td> </td>
<td>K</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>014</td>
<td>FF (NP form feed, new page)</td>
<td>44</td>
<td>2C</td>
<td>054</td>
<td> </td>
<td>,</td>
<td>76</td>
<td>4C</td>
<td>114</td>
<td> </td>
<td>L</td>
</tr>
<tr>
<td>13</td>
<td>D</td>
<td>015</td>
<td>CR (carriage return)</td>
<td>45</td>
<td>2D</td>
<td>055</td>
<td> </td>
<td>-</td>
<td>77</td>
<td>4D</td>
<td>115</td>
<td> </td>
<td>M</td>
</tr>
<tr>
<td>14</td>
<td>E</td>
<td>016</td>
<td>SO (shift out)</td>
<td>46</td>
<td>2E</td>
<td>056</td>
<td> </td>
<td>.</td>
<td>78</td>
<td>4E</td>
<td>116</td>
<td> </td>
<td>N</td>
</tr>
<tr>
<td>15</td>
<td>F</td>
<td>017</td>
<td>SI (shift in)</td>
<td>47</td>
<td>2F</td>
<td>057</td>
<td> </td>
<td>/</td>
<td>79</td>
<td>4F</td>
<td>117</td>
<td> </td>
<td>O</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>020</td>
<td>DLE (data link escape)</td>
<td>48</td>
<td>30</td>
<td>060</td>
<td> </td>
<td>0</td>
<td>80</td>
<td>50</td>
<td>120</td>
<td> </td>
<td>P</td>
</tr>
<tr>
<td>17</td>
<td>11</td>
<td>021</td>
<td>DC1 (device control 1)</td>
<td>49</td>
<td>31</td>
<td>061</td>
<td> </td>
<td>1</td>
<td>81</td>
<td>51</td>
<td>121</td>
<td> </td>
<td>Q</td>
</tr>
<tr>
<td>18</td>
<td>12</td>
<td>022</td>
<td>DC2 (device control 2)</td>
<td>50</td>
<td>32</td>
<td>062</td>
<td> </td>
<td>2</td>
<td>82</td>
<td>52</td>
<td>122</td>
<td> </td>
<td>R</td>
</tr>
<tr>
<td>19</td>
<td>13</td>
<td>023</td>
<td>DC3 (device control 3)</td>
<td>51</td>
<td>33</td>
<td>063</td>
<td> </td>
<td>3</td>
<td>83</td>
<td>53</td>
<td>123</td>
<td> </td>
<td>S</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>024</td>
<td>DC4 (device control 4)</td>
<td>52</td>
<td>34</td>
<td>064</td>
<td> </td>
<td>4</td>
<td>84</td>
<td>54</td>
<td>124</td>
<td> </td>
<td>T</td>
</tr>
<tr>
<td>21</td>
<td>15</td>
<td>025</td>
<td>NAK (negative acknowledge)</td>
<td>53</td>
<td>35</td>
<td>065</td>
<td> </td>
<td>5</td>
<td>85</td>
<td>55</td>
<td>125</td>
<td> </td>
<td>U</td>
</tr>
<tr>
<td>22</td>
<td>16</td>
<td>026</td>
<td>SYN (synchronous idle)</td>
<td>54</td>
<td>36</td>
<td>066</td>
<td> </td>
<td>6</td>
<td>86</td>
<td>56</td>
<td>126</td>
<td> </td>
<td>V</td>
</tr>
<tr>
<td>23</td>
<td>17</td>
<td>027</td>
<td>ETB (end of trans. block)</td>
<td>55</td>
<td>37</td>
<td>067</td>
<td> </td>
<td>7</td>
<td>87</td>
<td>57</td>
<td>127</td>
<td> </td>
<td>W</td>
</tr>
<tr>
<td>24</td>
<td>18</td>
<td>030</td>
<td>CAN (cancel)</td>
<td>56</td>
<td>38</td>
<td>070</td>
<td> </td>
<td>8</td>
<td>88</td>
<td>58</td>
<td>130</td>
<td> </td>
<td>X</td>
</tr>
<tr>
<td>25</td>
<td>19</td>
<td>031</td>
<td>EM (end of medium)</td>
<td>57</td>
<td>39</td>
<td>071</td>
<td> </td>
<td>9</td>
<td>89</td>
<td>59</td>
<td>131</td>
<td> </td>
<td>Y</td>
</tr>
<tr>
<td>26</td>
<td>1A</td>
<td>032</td>
<td>SUB (substitute)</td>
<td>58</td>
<td>3A</td>
<td>072</td>
<td> </td>
<td>;</td>
<td>90</td>
<td>5A</td>
<td>132</td>
<td> </td>
<td>Z</td>
</tr>
<tr>
<td>27</td>
<td>1B</td>
<td>033</td>
<td>ESC (escape)</td>
<td>59</td>
<td>3B</td>
<td>073</td>
<td> </td>
<td>&lt;</td>
<td>91</td>
<td>5B</td>
<td>133</td>
<td> </td>
<td>[</td>
</tr>
<tr>
<td>28</td>
<td>1C</td>
<td>034</td>
<td>FS (file separator)</td>
<td>60</td>
<td>3C</td>
<td>074</td>
<td> </td>
<td>&gt;</td>
<td>92</td>
<td>5C</td>
<td>134</td>
<td> </td>
<td>]</td>
</tr>
<tr>
<td>29</td>
<td>1D</td>
<td>035</td>
<td>GS (group separator)</td>
<td>61</td>
<td>3D</td>
<td>075</td>
<td> </td>
<td>5</td>
<td>93</td>
<td>5D</td>
<td>135</td>
<td> </td>
<td>{</td>
</tr>
<tr>
<td>30</td>
<td>1E</td>
<td>036</td>
<td>RS (record separator)</td>
<td>62</td>
<td>3E</td>
<td>076</td>
<td> </td>
<td>6</td>
<td>94</td>
<td>5E</td>
<td>136</td>
<td> </td>
<td>^</td>
</tr>
<tr>
<td>31</td>
<td>1F</td>
<td>037</td>
<td>US (unit separator)</td>
<td>63</td>
<td>3F</td>
<td>077</td>
<td> </td>
<td>7</td>
<td>95</td>
<td>5F</td>
<td>137</td>
<td> </td>
<td>_</td>
</tr>
</tbody>
</table>

Source: [www.LookupTables.com](http://www.LookupTables.com)
## Extended ASCII Table

<table>
<thead>
<tr>
<th>Code</th>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>Ç</td>
<td>144</td>
<td>É</td>
<td>160</td>
<td>á</td>
<td>176</td>
<td></td>
</tr>
<tr>
<td>129</td>
<td>ü</td>
<td>145</td>
<td>æ</td>
<td>161</td>
<td>í</td>
<td>177</td>
<td></td>
</tr>
<tr>
<td>130</td>
<td>é</td>
<td>146</td>
<td>Æ</td>
<td>162</td>
<td>ó</td>
<td>178</td>
<td></td>
</tr>
<tr>
<td>131</td>
<td>â</td>
<td>147</td>
<td>ò</td>
<td>163</td>
<td>ú</td>
<td>179</td>
<td></td>
</tr>
<tr>
<td>132</td>
<td>ä</td>
<td>148</td>
<td>ö</td>
<td>164</td>
<td>ñ</td>
<td>180</td>
<td></td>
</tr>
<tr>
<td>133</td>
<td>à</td>
<td>149</td>
<td>ò</td>
<td>165</td>
<td>ñ</td>
<td>181</td>
<td></td>
</tr>
<tr>
<td>134</td>
<td>å</td>
<td>150</td>
<td>ü</td>
<td>166</td>
<td>À</td>
<td>182</td>
<td></td>
</tr>
<tr>
<td>135</td>
<td>ç</td>
<td>151</td>
<td>û</td>
<td>167</td>
<td>û</td>
<td>183</td>
<td></td>
</tr>
<tr>
<td>136</td>
<td>ê</td>
<td>152</td>
<td>ÿ</td>
<td>168</td>
<td>ë</td>
<td>184</td>
<td></td>
</tr>
<tr>
<td>137</td>
<td>è</td>
<td>153</td>
<td>Ö</td>
<td>169</td>
<td>Ô</td>
<td>185</td>
<td></td>
</tr>
<tr>
<td>138</td>
<td>ê</td>
<td>154</td>
<td>Ü</td>
<td>170</td>
<td>Ñ</td>
<td>186</td>
<td></td>
</tr>
<tr>
<td>139</td>
<td>ë</td>
<td>155</td>
<td>Ï</td>
<td>171</td>
<td>½</td>
<td>187</td>
<td></td>
</tr>
<tr>
<td>140</td>
<td>å</td>
<td>156</td>
<td>£</td>
<td>172</td>
<td>¼</td>
<td>188</td>
<td></td>
</tr>
<tr>
<td>141</td>
<td>ì</td>
<td>157</td>
<td>¥</td>
<td>173</td>
<td>i</td>
<td>189</td>
<td></td>
</tr>
<tr>
<td>142</td>
<td>Ä</td>
<td>158</td>
<td>€</td>
<td>174</td>
<td>«</td>
<td>190</td>
<td></td>
</tr>
<tr>
<td>143</td>
<td>Â</td>
<td>159</td>
<td>¥</td>
<td>175</td>
<td>»</td>
<td>191</td>
<td></td>
</tr>
</tbody>
</table>

Source: [www.LookupTables.com](http://www.LookupTables.com)
Casting

• Casting is a method to correctly use variables of different types together
• It allows to treat a variable of one type as if it were of another type in a specific context
• When it makes sense, the compiler does it for us automatically

• Implicit (automatic)
  ```
  int x = 1;
  float y = 2.3;
  x = x + y;
  ```

  x= 3 compiler automatically casted (=converted) y to be an integer just for this instruction

• Explicit (non-automatic)
  ```
  char c = ‘A’ ;
  int x = (int) c;
  ```

  Explicit casting from char to int. The value of x here is 65
Operators

• Assignment =

• Arithmetic * / % + -

• Increment ++ -- += -=

• Relational < <= > >= == !=

• Logical && || !

• Bitwise & | ~ ^ << >>

• Comma ,
Operators – Assignment

```c
int x = 3;
x = 7;

int x, y = 5;
x = y = 7;
float y = 2.3, z = 3, q = 700;

int i, j, k;
k = (i=2, j=3);
printf("i = %d, j = %d, k = %d\n", i, j, k);
```

The comma operator allows us to perform multiple assignments/declarations.
Operators - Arithmetic

- Arithmetic operators have a **precedence**
  ```c
  int x;
  x = 3 + 5 * 2 - 4 / 2;
  ```

- We can use parentheses () to impose our precedence order
  ```c
  int x;
  x = (3 + 5) * (2 - 4) / 2;
  ```

- `%` returns the module (or the remainder of the division)
  ```c
  int x;
  x = 5 % 3;  // x = 2
  ```

- We have to be careful with integer vs. float division : remember automatic casting!
  ```c
  int x = 3;
  float y;
  y = x / 2;  // y = 1.00
  ```
Operators - Arithmetic

- Arithmetic operators have a precedence
  ```c
  int x;
  x = 3 + 5 * 2 - 4 / 2;
  ```

- We can use parentheses () to impose our precedence order
  ```c
  int x;
  x = (3 + 5) * (2 - 4) / 2;
  ```

- % returns the module (or the remainder of the division)
  ```c
  int x;
  x = 5 % 3; // x = 2
  ```

- We have to be careful with integer vs. float division: remember automatic casting!
  ```c
  int x = 3;
  float y;
  y = x / 2; // y = 1.00
  ```

Possible fixes:
1) `float x = 3;
y = 1 / 2; // y = 0.00`
2) `y = (float) x / 2; Then y = 1.50`

Possible fix: `y = 1.0/2; Then y = 0.50`
Operators - Increment

```
int x = 3, y, z;

x++;  // x is incremented at the end of statement
++x;  // x is incremented at the beginning of statement

y = ++x + 3;  // x = x + 1; y = x + 3;

z = x++ + 3;  // z = x + 3; x = x + 1;

x -= 2;  // x = x - 2;
```
Operators - Relational

- Return 0 if statement is false, 1 if statement is true

```c
int x = 3, y = 2, z, k, t;

z = x > y;       // z = 1
k = x <= y;      // k = 0
t = x != y;      // t = 1
```
Operators - Logical

- A variable with value 0 is false, a variable with value $!=0$ is true

```c
int x = 3, y = 0, z, k, t, q = -3;

z = x && y;    // z = 0;        x is true but y is false
k = x || y;    // k = 1;        x is true

q = !q;        // q is true
```
Review: Operators - Bitwise

- Work on the binary representation of data
- Remember: computers store and see data in binary format!

```c
int x, y, z, t, q, s, v;

x = 3;            00000000000000000000000000000011
y = 16;           00000000000000000000000000000100

z = x << 1;  equivalent to  z = x \cdot 2^1 00000000000000000000000000000110

x = y >> 3;  equivalent to  t = y \cdot 2^{-3} 00000000000000000000000000000010

t = x & y;       00000000000000000000000000000000

s = x | y;       00000000000000000000000000000101

v = x ^ y;  \downarrow \text{XOR}  00000000000000000000000000000101
```
• **printf** is a function used to print to standard output (command line)

• **Syntax:**
  ```c
  printf("format1 format2 ...", variable1, variable2,...);
  ```

• **Format characters:**
  - `%d` or `%i` integer
  - `%f` float
  - `%lf` double
  - `%c` char
  - `%u` unsigned
  - `%s` string

```
  Format
  % 0 n1 . n2 t
  ```
  - pad with zeros (optional)
  - type
  - number of digits after the decimal point
  - number of digits before the decimal point
#include <stdio.h>

int main() {

    int a, b;
    float c, d;
    a = 15;
    b = a / 2;

    printf("%d\n", b);
    printf("%3d\n", b);
    printf("%03d\n", b);

    c = 15.3;
    d = c / 3;
    printf("%3.2f\n", d);

    return(0);
}

Output:

7
7
007

5.10
printf

Escape sequences

\n      newline
\t      tab
\v      vertical tab
\f      new page
\b      backspace
\r      carriage return
Assignment

- Read PCP Chapter 3 and 4