Reading multiple characters using scanf

- If multiple characters are read using scanf("%c", &c)
  # include <stdio.h>
  int main ()
  {
    char a,b,c;
    printf ("Enter first character 
");
    scanf ("%c", &a);
    printf ("Enter second character 
");
    scanf ("%c", &b);
    printf ("Enter third character 
");
    scanf ("%c", &c);
    printf ("%c
%c
%c
", a, b, c);
  }

- Does not work as “Enter” is read as a character

Reading multiple characters using scanf (cont.)

- “Enter” is not printed
- The integer value mapped to “Enter” can be printed
  # include <stdio.h>
  int main ()
  {
    char a,b,c;
    printf ("Enter first character 
");
    scanf ("%c", &a);
    printf ("Enter second character 
");
    scanf ("%c", &b);
    printf ("Enter third character 
");
    scanf ("%c", &c);
    printf ("%d
%d
%d
", a, b, c); /*The integer value associated with the characters is printed*/
  }
Reading multiple characters using scanf (cont.)

- Read all characters at one go, and press "Enter" only at the end
- # include <stdio.h>
- int main()
  
  char a, b, c;
  printf(" Enter three characters \n");
  scanf("%c%c%c", &a,&b,&c);
  printf("%c
%c
%c
", a, b, c);
  printf("%d
%d
%d
", a, b, c);

Reading variable number of characters

- A loop needs to be used for variable number of characters
- Input all characters at a time and then press "Enter"
- int i, n;
  char c;
  printf(" Enter the number of characters \n");
  scanf("%d", &n);
  for (i = 0; i < n; i++)
    
    scanf("%c", &c);
    printf("%c\n", c); //printed only after all characters are read
    printf("%d\n", c); //printed only after all characters are read
  } //i has the value of the last character

Reading with scanf: Recap

- scanf requires "Enter" before it can read
- scanf("%c", &c) – reads blank, “tab” or “Enter” as a character
- scanf("%c", &c) – ignores all whitespace and reads the next non-whitespace character
- scanf("%s", name) – reads a sequence of characters as a string
  - Does not read in whitespace characters
  - Can store more characters than the length of the character array, which is an array boundary overflow problem
  - Array boundary overflow may cause undetermined bugs
  - Has a null character at the end of the string

Reading with scanf: Recap

- scanf requires "Enter" before it can read
- scanf("%c", &c) – reads blank, “tab” or “Enter” as a character
- scanf("%c", &c) – ignores all whitespace and reads the next non-whitespace character
- scanf("%s", name) – reads a sequence of characters as a string
  - Does not read in whitespace characters
  - Can store more characters than the length of the character array, which is an array boundary overflow problem
  - Array boundary overflow may cause undetermined bugs
  - Has a null character at the end of the string
## Review of scanf

<table>
<thead>
<tr>
<th>Declaration</th>
<th>Statement</th>
<th>Input (□ means blank)</th>
<th>Value stored</th>
</tr>
</thead>
<tbody>
<tr>
<td>char c</td>
<td>scanf(&quot;%c&quot;, &amp;c);</td>
<td>□ g □</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>\n \n</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>scanf(&quot;%c&quot;, &amp;c);</td>
<td>□ g g</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>\n\t\n a</td>
<td>A</td>
</tr>
</tbody>
</table>

### getchar() for reading characters

- getchar() reads a single character
- `int i = 0; char c; while (c = getchar (), c !='\n') // Stop input with Enter`  
  ```c
  { //comma operator: with final value being rightmost expression  
    printf("%c", c);  
    i++;  
  }
  ```
- `printf(" Number of characters input is %d\n", i);`  
- `Number of characters include blank and tab`  
- `Reads in input till "Enter" is pressed`

### putchar() for printing characters

- `putchar(c)` prints the character `c`
- `char c = 't'; char c; int i = 0; while (c = getchar (), c != '\n') // Stop input with Enter`  
  ```c
  { //comma operator: with final value being rightmost expression  
    putchar(c);  
    i++;  
  }
  ```
- `printf(" Number of characters input is %d\n", i);`

### Displaying special characters using printf

<table>
<thead>
<tr>
<th>Character</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>\n</td>
<td>New line or &quot;Enter&quot;</td>
</tr>
<tr>
<td>\t</td>
<td>Horizontal Tab (horizontal spaces)</td>
</tr>
<tr>
<td>'</td>
<td>Single quote</td>
</tr>
<tr>
<td>&quot;</td>
<td>Double quote</td>
</tr>
<tr>
<td>\</td>
<td>Backslash</td>
</tr>
<tr>
<td>\v</td>
<td>Vertical tab (vertical space)</td>
</tr>
<tr>
<td>\b</td>
<td>backspace (moves the cursor back one character)</td>
</tr>
<tr>
<td>\r</td>
<td>Carriage return (moves cursor to beginning of line)</td>
</tr>
<tr>
<td>\a</td>
<td>audible alert or bell</td>
</tr>
<tr>
<td>?</td>
<td>question mark</td>
</tr>
<tr>
<td>\0</td>
<td>null character</td>
</tr>
</tbody>
</table>
Output formatting with printf

<table>
<thead>
<tr>
<th>Value</th>
<th>Placeholder</th>
<th>Output ( [] means blank )</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>l='a'</td>
<td>printf(&quot;%c&quot;,l);</td>
<td>a</td>
<td>display in 3 columns, right justified</td>
</tr>
<tr>
<td></td>
<td>printf(&quot;%3c&quot;,l);</td>
<td>a</td>
<td>display in 3 columns, right justified</td>
</tr>
<tr>
<td></td>
<td>printf(&quot;%-3c&quot;,l);</td>
<td>a</td>
<td>display in 3 columns, left justified</td>
</tr>
<tr>
<td></td>
<td>printf(&quot;%4c&quot;,l);</td>
<td>a</td>
<td>display in variable number of columns, specified here as 4</td>
</tr>
<tr>
<td></td>
<td>printf(&quot;%-4c&quot;,l);</td>
<td>a</td>
<td>display in variable number of columns, specified here as -4, i.e. left justification</td>
</tr>
</tbody>
</table>

Sample program on formatting characters

- Display a sequence of numbers from 1 to 5, in a diagonal format as below.
  1
  2
  3
  4
  5

```c
char c = '1';
for (j = 1; j <= 5; j++)
{
    printf("%c", c+j-1);
}
```

Output formatting with printf

<table>
<thead>
<tr>
<th>Value</th>
<th>Placeholder</th>
<th>Output ( [] means blank )</th>
<th>Character format:</th>
</tr>
</thead>
<tbody>
<tr>
<td>-10</td>
<td>%d</td>
<td>-10</td>
<td>&quot;&quot;str[10]&quot;,&quot;ESc101&quot;;</td>
</tr>
<tr>
<td></td>
<td>%2d</td>
<td>-10</td>
<td>Character stored:</td>
</tr>
<tr>
<td></td>
<td>%4d</td>
<td>-10</td>
<td>Character format:</td>
</tr>
<tr>
<td></td>
<td>%4d</td>
<td>-10</td>
<td>&quot;&quot;str[10]&quot;,&quot;ESc101&quot;;</td>
</tr>
<tr>
<td>10</td>
<td>%04d</td>
<td>0010</td>
<td>Int format:</td>
</tr>
<tr>
<td>49.76</td>
<td>%.3f</td>
<td>49.760</td>
<td>Float format:</td>
</tr>
<tr>
<td></td>
<td>%.1f</td>
<td>49.7</td>
<td>Float format:</td>
</tr>
<tr>
<td></td>
<td>%10.2f</td>
<td>49.760</td>
<td>Float format:</td>
</tr>
<tr>
<td></td>
<td>%10.2f</td>
<td>49.760</td>
<td>String format:</td>
</tr>
<tr>
<td></td>
<td>%10.3e</td>
<td>4.976e+01</td>
<td>String format:</td>
</tr>
</tbody>
</table>

Variable spacing for display, etc.

- Character stored: ""str[10]","ESc101";  
- Character format: ""str[10]","ESc101";  
- Int format: 10"  
- Int format: 00010"  
- Float format: 49.760"  
- Float format: 000049.760"  
- String format: 8.5"