Lecture 6
ASCII Code

- Each character is assigned a unique integer value (code) between 32 and 127

- The code of a character is represented by an 8-bit unit.
  - Since an 8-bit unit can hold a total of $2^8=256$ values and the computer character set is much smaller than that, some values of this 8-bit unit do not correspond to visible characters

- But not a good idea to remember exact ASCII codes while programming. Use the facts that
  - C stores characters as integers
  - Ascii codes of some important characters are contiguous (digits, lowercase alphabets, uppercase alphabets)
<table>
<thead>
<tr>
<th>Decimal</th>
<th>Hex</th>
<th>Binary</th>
<th>Character</th>
<th>Decimal</th>
<th>Hex</th>
<th>Binary</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>20</td>
<td>00100000</td>
<td>SPACE</td>
<td>80</td>
<td>50</td>
<td>01010000</td>
<td>P</td>
</tr>
<tr>
<td>33</td>
<td>21</td>
<td>00100001</td>
<td>!</td>
<td>81</td>
<td>51</td>
<td>01010001</td>
<td>Q</td>
</tr>
<tr>
<td>34</td>
<td>22</td>
<td>00100010</td>
<td>&quot;</td>
<td>82</td>
<td>52</td>
<td>01010010</td>
<td>R</td>
</tr>
<tr>
<td>35</td>
<td>23</td>
<td>00100011</td>
<td>#</td>
<td>83</td>
<td>53</td>
<td>01010011</td>
<td>S</td>
</tr>
<tr>
<td>36</td>
<td>24</td>
<td>00100100</td>
<td>$</td>
<td>84</td>
<td>54</td>
<td>01010100</td>
<td>T</td>
</tr>
<tr>
<td>37</td>
<td>25</td>
<td>00100101</td>
<td>%</td>
<td>85</td>
<td>55</td>
<td>01010101</td>
<td>U</td>
</tr>
<tr>
<td>38</td>
<td>26</td>
<td>00100110</td>
<td>&amp;</td>
<td>86</td>
<td>56</td>
<td>01010110</td>
<td>V</td>
</tr>
<tr>
<td>39</td>
<td>27</td>
<td>00100111</td>
<td>'</td>
<td>87</td>
<td>57</td>
<td>01010111</td>
<td>W</td>
</tr>
<tr>
<td>40</td>
<td>28</td>
<td>00101000</td>
<td>(</td>
<td>88</td>
<td>58</td>
<td>01011000</td>
<td>X</td>
</tr>
<tr>
<td>41</td>
<td>29</td>
<td>00101001</td>
<td>)</td>
<td>89</td>
<td>59</td>
<td>01011001</td>
<td>Y</td>
</tr>
<tr>
<td>42</td>
<td>2a</td>
<td>00101010</td>
<td>*</td>
<td>90</td>
<td>5a</td>
<td>01011010</td>
<td>Z</td>
</tr>
<tr>
<td>43</td>
<td>2b</td>
<td>00101011</td>
<td>+</td>
<td>91</td>
<td>5b</td>
<td>01011101</td>
<td>[</td>
</tr>
<tr>
<td>44</td>
<td>2c</td>
<td>00101100</td>
<td>,</td>
<td>92</td>
<td>5c</td>
<td>01011100</td>
<td>\</td>
</tr>
<tr>
<td>45</td>
<td>2d</td>
<td>00101101</td>
<td>-</td>
<td>93</td>
<td>5d</td>
<td>01011101</td>
<td>]</td>
</tr>
<tr>
<td>46</td>
<td>2e</td>
<td>00101110</td>
<td>.</td>
<td>94</td>
<td>5e</td>
<td>01011110</td>
<td>^</td>
</tr>
<tr>
<td>47</td>
<td>2f</td>
<td>00101111</td>
<td>/</td>
<td>95</td>
<td>5f</td>
<td>01011111</td>
<td>_</td>
</tr>
<tr>
<td>48</td>
<td>30</td>
<td>00110000</td>
<td>0</td>
<td>96</td>
<td>60</td>
<td>01100000</td>
<td>`</td>
</tr>
<tr>
<td>49</td>
<td>31</td>
<td>00110001</td>
<td>1</td>
<td>97</td>
<td>61</td>
<td>01100001</td>
<td>a</td>
</tr>
<tr>
<td>50</td>
<td>32</td>
<td>00110010</td>
<td>2</td>
<td>98</td>
<td>62</td>
<td>01100100</td>
<td>b</td>
</tr>
<tr>
<td>Unicode</td>
<td>Code_Point</td>
<td>Binary</td>
<td>Decimal</td>
<td>Hexadecimal</td>
<td>Character</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>------------</td>
<td>--------</td>
<td>---------</td>
<td>-------------</td>
<td>-----------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>51</td>
<td>33</td>
<td>00110011</td>
<td>3</td>
<td>99</td>
<td>c</td>
<td></td>
<td></td>
</tr>
<tr>
<td>52</td>
<td>34</td>
<td>00110100</td>
<td>4</td>
<td>100</td>
<td>d</td>
<td></td>
<td></td>
</tr>
<tr>
<td>53</td>
<td>35</td>
<td>00110101</td>
<td>5</td>
<td>101</td>
<td>e</td>
<td></td>
<td></td>
</tr>
<tr>
<td>54</td>
<td>36</td>
<td>00110110</td>
<td>6</td>
<td>102</td>
<td>f</td>
<td></td>
<td></td>
</tr>
<tr>
<td>55</td>
<td>37</td>
<td>00110111</td>
<td>7</td>
<td>103</td>
<td>g</td>
<td></td>
<td></td>
</tr>
<tr>
<td>56</td>
<td>38</td>
<td>00111000</td>
<td>8</td>
<td>104</td>
<td>h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>57</td>
<td>39</td>
<td>00111001</td>
<td>9</td>
<td>105</td>
<td>i</td>
<td></td>
<td></td>
</tr>
<tr>
<td>58</td>
<td>3a</td>
<td>00111010</td>
<td>:</td>
<td>106</td>
<td>j</td>
<td></td>
<td></td>
</tr>
<tr>
<td>59</td>
<td>3b</td>
<td>00111011</td>
<td>;</td>
<td>107</td>
<td>k</td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>3c</td>
<td>00111100</td>
<td>&lt;</td>
<td>108</td>
<td>l</td>
<td></td>
<td></td>
</tr>
<tr>
<td>61</td>
<td>3d</td>
<td>00111101</td>
<td>=</td>
<td>109</td>
<td>m</td>
<td></td>
<td></td>
</tr>
<tr>
<td>62</td>
<td>3e</td>
<td>00111110</td>
<td>&gt;</td>
<td>110</td>
<td>n</td>
<td></td>
<td></td>
</tr>
<tr>
<td>63</td>
<td>3f</td>
<td>00111111</td>
<td>?</td>
<td>111</td>
<td>o</td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>40</td>
<td>01000000</td>
<td>@</td>
<td>112</td>
<td>p</td>
<td></td>
<td></td>
</tr>
<tr>
<td>65</td>
<td>41</td>
<td>01000001</td>
<td>A</td>
<td>113</td>
<td>q</td>
<td></td>
<td></td>
</tr>
<tr>
<td>66</td>
<td>42</td>
<td>01000010</td>
<td>B</td>
<td>114</td>
<td>r</td>
<td></td>
<td></td>
</tr>
<tr>
<td>67</td>
<td>43</td>
<td>01000011</td>
<td>C</td>
<td>115</td>
<td>s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>68</td>
<td>44</td>
<td>01000100</td>
<td>D</td>
<td>116</td>
<td>t</td>
<td></td>
<td></td>
</tr>
<tr>
<td>69</td>
<td>45</td>
<td>01000101</td>
<td>E</td>
<td>117</td>
<td>u</td>
<td></td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>46</td>
<td>01000110</td>
<td>F</td>
<td>118</td>
<td>v</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>71</td>
<td>47</td>
<td>01000111</td>
<td>G</td>
<td>119</td>
<td>77</td>
<td>01110111</td>
<td>w</td>
</tr>
<tr>
<td>72</td>
<td>48</td>
<td>01001000</td>
<td>H</td>
<td>120</td>
<td>78</td>
<td>01111000</td>
<td>x</td>
</tr>
<tr>
<td>73</td>
<td>49</td>
<td>01001001</td>
<td>I</td>
<td>121</td>
<td>79</td>
<td>01111001</td>
<td>y</td>
</tr>
<tr>
<td>74</td>
<td>4a</td>
<td>01001010</td>
<td>J</td>
<td>122</td>
<td>7a</td>
<td>01111010</td>
<td>z</td>
</tr>
<tr>
<td>75</td>
<td>4b</td>
<td>01001011</td>
<td>K</td>
<td>123</td>
<td>7b</td>
<td>01111011</td>
<td>{</td>
</tr>
<tr>
<td>76</td>
<td>4c</td>
<td>01001100</td>
<td>L</td>
<td>124</td>
<td>7c</td>
<td>01111100</td>
<td></td>
</tr>
<tr>
<td>77</td>
<td>4d</td>
<td>01001101</td>
<td>M</td>
<td>125</td>
<td>7d</td>
<td>01111101</td>
<td>}</td>
</tr>
<tr>
<td>78</td>
<td>4e</td>
<td>01001110</td>
<td>N</td>
<td>126</td>
<td>7e</td>
<td>01111110</td>
<td>~</td>
</tr>
<tr>
<td>79</td>
<td>4f</td>
<td>01001111</td>
<td>O</td>
<td>127</td>
<td>7f</td>
<td>01111111</td>
<td>DELETE</td>
</tr>
</tbody>
</table>
### Quiz...

<table>
<thead>
<tr>
<th>Expression</th>
<th>Value?</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘9’ &gt;= ‘0’</td>
<td>1 (true)</td>
</tr>
<tr>
<td>‘a’ &lt; ‘e’</td>
<td>1 (true)</td>
</tr>
<tr>
<td>‘Z’ == ‘z’</td>
<td>0 (false)</td>
</tr>
<tr>
<td>‘a’ &lt;= ‘A’</td>
<td>0 (false)</td>
</tr>
</tbody>
</table>
Example: checking if a character is a lowercase alphabet

```c
int main()
{
    /* Read a character and display whether it is lower case or upper case */
    char c1;
    scanf("%c", &c1);
    /* the ascii code of c1 must lie between the
    ascii codes of ‘a’ and ‘z’ */
    if (c1 >= ‘a’ && c1<= ‘z’)
        printf("%c is a lowercase alphabet\n", c1);
    else printf("%c is not a lowercase alphabet\n", c1);
    return 0;
}
```
Example: converting a character from lowercase to uppercase

```c
int main()
{
    char c1;
    scanf("%c", &c1);
    /* convert to uppercase if lowercase, else leave as it is */
    if (c1 >= 'a' && c1 <= 'z')
        /* since ascii codes of uppercase letters are contiguous, the
         uppercase version of c1 will be as far away from the ascii code
         of ‘A’ as it is from the ascii code of ‘a’ */
        c1 = 'A' + (c1 - 'a');
    printf("The letter is %c\n", c1);
    return 0;
}
```
Exercise

- Write a program that:
  - When the user enters a or A, displays “First letter”
  - When the user enters z or Z, displays “last letter”.
  - For any other letter entered by the user it displays “middle letter”.
Switching with char type

cchar letter;
scanf("%c", &letter);
switch ( letter ) {
    case 'A':
        printf ("First letter \n");
        break;
    case 'Z':
        printf ("Last letter \n");
        break;
    default :
        printf ("Middle letter \n");
}
Switching with char type

char letter;
scanf("%c", &letter);
switch ( letter ) {
    case 'A':
    case 'a':
        printf ("First letter \n");
        break;
    case 'Z':
    case 'z':
        printf ("Last letter \n");
        break;
    default :
        printf ("Middle letter \n");
}
char letter;
scanf("%c", &letter);
switch ( letter ) {
    case 'A':
        printf ("First letter \n");
        break;
    case 'Z':
        printf ("Last letter \n");
        break;
    default :
        printf ("Middle letter \n");
        printf ("Will print this statement for all letters other than A or Z");
}
Another Example

switch (choice = getchar()) {
    case ‘r’ :
    case ‘R’ : printf(“Red”);
        break;
    case ‘b’ :
    case ‘B’ : printf(“Blue”);
        break;
    case ‘g’ :
    case ‘G’ : printf(“Green”);
        break;
    default: printf(“Black”);
}

Another Example

```c
switch (choice = getchar()) {
    case 'r':
    case 'R': printf("Red");
        break;
    case 'b':
    case 'B': printf("Blue");
        break;
    case 'g':
    case 'G': printf("Green");
        break;
    default: printf("Black");
}
```

Since there isn’t a break statement here, the control passes to the next statement (printf) without checking the next condition.
int main () {
    int operand1, operand2;
    int result = 0;
    char operation ;

    /* Get the input values */
    printf ("Enter operand1 :");
    scanf("%d",&operand1) ;
    printf ("Enter operation :");
    scanf ("\n%c",&operation);
    printf ("Enter operand 2 :");
    scanf ("%d", &operand2);
    switch (operation)  {
    case '+':
        result=operand1+operand2;
        break;
    case '-':
        result=operand1-operand2;
        break;
    case '*':
        result=operand1*operand2;
        break;
    case '/':
        if (operand2 !=0)
            result=operand1/operand2;
        else
            printf("Divide by 0 error");
        break;
    default:
        printf("Invalid operation\n");
        return;
    }
    printf ("The answer is %d\n",result);
    return 0;
}
Practice Problems

1. Read in 3 integers and print a message if any one of them is equal to the sum of the other two.

2. Read in the coordinates of two points and print the equation of the line joining them in $y = mx + c$ form.

3. Read in the coordinates of 3 points in 2-d plane and check if they are collinear. Print a suitable message.

4. Read in the coordinates of a point, and the center and radius of a circle. Check and print if the point is inside or outside the circle.

5. Read in the coefficients $a$, $b$, $c$ of the quadratic equation $ax^2 + bx + c = 0$, and print its roots nicely (for imaginary roots, print in $x + iy$ form)

6. Suppose the digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 are mapped to the lowercase letters a, b, c, d, e, f, g, h, i, j respectively. Read in a single digit integer as a character (using %c in scanf) and print its corresponding lowercase letter. Do this both using switch and without using switch (two programs). Do not use any ascii code value directly.

7. Suppose that you have to print the grades of a student, with $\geq 90$ marks getting EX, 80-89 getting A, 70-79 getting B, 60-69 getting C, 50-59 getting D, 35-49 getting P and $<30$ getting F. Read in the marks of a student and print his/her grade.
Lecture 7
Looping
Loops

- Group of statements that are executed repeatedly while some condition remains true.
- Each execution of the group of statements is called an iteration of the loop.
- Examples:
  - Keep on dividing a number by two and display the remainder until the number becomes 1 or 0.
  - Multiply a number with itself n times.
  - Keep on reading a number from keyboard and adding, until the user enters 0.
Exam

counter ← 1, sum ← 0

counter < 6

false
t

t

t

t

t

t

t

t

t

t

t

t

t

true

input n

sum ← sum + n

counter = counter + 1

output sum

Example

Read 5 integers and display the their sum
Example

Read exam marks as input, display the appropriate message based on the rules below:

- If marks is greater than 49, display “PASS”, otherwise display “FAIL”
- However, for input outside the 0-100 range, display “WRONG INPUT” and prompt the user to input again until a valid input is entered…
Input m

\[ m < 0 \text{ or } m > 100 \]

If false, proceed to “WRONG INPUT”

If true, continue with input m

\[ m > 49 \]

If true, “PASS”

If false, “FAIL”
input m

m<0 || m>100

“WRONG INPUT”

input m

m>49

“PASS”

“FAIL”
Types of Loops

- Loops are controlled by boolean expressions
- C has three kinds of loops:
  - `while loop`
  - `do loop`
  - `for loop`
The condition to be tested is a logical expression enclosed in parentheses. The expression is evaluated, and if its value is non-zero, the statement is executed. Then the expression is evaluated again and the same thing repeats. The loop terminates when the expression evaluates to 0.
Looping: **while** statement

```
while (expression) {
    statement;
    Block of statements;
}
```
Looping: **while** statement

```java
while (expression) {
    Block of statements;
}
```

The condition to be tested is any expression enclosed in parentheses. The expression is evaluated, and if its value is non-zero, the statement is executed. Then the expression is evaluated again and the same thing repeats. The loop **terminates** when the expression evaluates to 0.
```c
int main()
{
    int i = 1, n;
    scanf("%d", &n);
    while (i <= n) {
        printf("Line no : %d\n", i);
        i = i + 1;
    }
}
```
Example: Exercise Till Lose Weight

```c
int weight;
scanf("%d", &weight);
printf("Weight is: %d\n", weight);
while ( weight > 65 ) {
    printf("Go, exercise, ");
    printf("then come back... \n");
    printf("Measure and Enter your weight: ");
    scanf("%d", &weight);
}
```
int main() {
    int N, count, sum;
    scanf("%d", &N);
    sum = 0;
    count = 1;
    while (count <= N) {
        sum = sum + count;
        count = count + 1;
    }
    printf("Sum = %d\n", sum);
    return 0;
}
SUM = $1^2 + 2^2 + 3^2 + \ldots + N^2$

```c
int main() {
    int N, count, sum;
    scanf ("%d", &N);
    sum = 0;
    count = 1;
    while (count <= N) {
        sum = sum + count * count;
        count = count + 1;
    }
    printf ("Sum = %d\n", sum);
    return 0;
}
```
Suppose your Rs 10000 is earning interest at 1% per month. How many months for your money to double?

```c
int main() {
    double my_money = 10000.0;
    int n = 0;
    while (my_money < 20000.0) {
        my_money = my_money * 1.01;
        n++;
    }
    printf ("My money will double in %d months.\n", n);
    return 0;
}
```
int main() {
    double max = 0.0, next;
    printf ("Enter positive numbers only, end with 0 or a negative number\n");
    scanf("%lf", &next);
    while (next > 0) {
        if (next > max) max = next;
        scanf("%lf", &next);
    }
    printf ("The maximum number is %lf\n", max) ;
    return 0;
}
Enter positive numbers only, end with 0 or a negative number
45
32
7
5
0
The maximum number is 45.000000
Find the sum of digits of a number

```c
int main(){
    int n, sum=0;
    scanf ("%d", &n);
    while (n != 0) {
        sum = sum + (n % 10);
        n = n / 10;
    }
    printf ("The sum of digits is %d \n", sum);
    return 0;
}
```

Output
573254
The sum of digits is 26
Compute GCD of two numbers

```c
int main() {
    int A, B, temp;
    scanf(“%d %d”, &A, &B);
    if (A > B) {
        temp = A; A = B; B = temp;
    }
    while ((B % A) != 0) {
        temp = B % A;
        B = A; Euclid’s Algorithm
        A = temp;
    }
    printf(“The GCD is %d”, A);
    return 0;
}
```

Initial: A=12, B=45
Iteration 1: temp=9, B=12, A=9
Iteration 2: temp=3, B=9, A=3
B % A = 0  ➔ GCD is 3
Exercise

- Write a program to determine the most significant digit of the value stored in an integer variable num.

- For example:
  - If num=457138, your program should display 4.
Find the most significant digit of a number

```c
int main()
{
    int n, msdigit=0;
    scanf ("%d", &n);
    while (n != 0)
    {
        msdigit = n % 10;
        n = n / 10;
    }
    printf ("Most significant digit is %d \n", msdigit);
    return 0;
}
```
Lecture 8
Loopping: \texttt{for} Statement

- Most commonly used looping structure in C

\begin{verbatim}
for ( expr1; expr2; expr3)
  statement;

for ( expr1; expr2; expr3)
  Block of statements;
\end{verbatim}

\begin{tabular}{|l|}
\hline
expr1 (init) : initialize parameters  \\
expr2 (test): test condition, loop continues if expression is non-0  \\
expr3 (update): used to alter the value of the parameters after each iteration  \\
statement (body): body of loop  \\
\hline
\end{tabular}
For loop has three parts:

- **initial value** of the control variable.
- **condition** that tests whether control variable has reached the desired value.
- **increment (or decrement)** the control variable.

```c
for ( initialization; loop condition; loop update )
{
    // loop body
}
```
for ( expr1; expr2; expr3)  
statement;

for ( expr1; expr2; expr3){
    Block of statements;
}

expr1  
(init)

expr2  
(test)

statement  
(body)

expr3  
(update)

True

False
**First Example: Display 1 to 10**

```c
int counter = 1;    /* initialization */
while (counter <= 10) {    /* repetition condition*/
    printf("%d\n", counter);
    ++counter;    /* increment */
}
```

**While Loop Version**

```c
int counter;
for (counter=1;counter<=10;counter++)
    printf("%d\n",counter);
```

**For Loop Version**
Example 2: Compute Factorial

```c
int main () {
    int N, count, prod;
    scanf (“%d”, &N) ;
    prod = 1;
    for (count = 1;count <= N; ++count)
        prod = prod * count;
    printf (“Factorial = %d\n”, prod) ;
    return 0;
}
```

Output

```
7
Factorial = 5040
```
Computing $e^x$ series up to N terms

```c
int main () {
    float x, term, sum;
    int n, count;
    scanf ("%f", &x);
    scanf ("%d", &n);
    term = 1.0; sum = 0;
    for (count = 1; count <= n; ++count)  {
        sum += term;
        term *= x/count;
    }
    printf ("%f\n", sum);
    return 0;
}
```

The series sum is 7.506626

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$
Computing $e^x$ series up to 4 decimal places

```c
int main() {
    float x, term, sum;
    int cnt;
    scanf (“%f”, &x) ;
    term = 1.0; sum = 0;
    for (cnt = 1; term >= 0.0001; ++cnt) {
        sum += term;
        term *= x/cnt;
    }
    printf (“%f
”, sum);
    return 0;
}
```
The comma operator

We can give several statements separated by commas in an expression.

```c
for (fact=1, i=1; i<=10; i++)
    fact = fact * i;
```

```c
for (sum=0, i=1; i<=N, i++)
    sum = sum + i * i;
```
Equivalence of `for` and `while`

`for ( expr1; expr2; expr3)  
statement;`

Same as

`expr1;  
while (expr2)  
{  
  statement;  
  expr3;  
}`
Sum of first N Natural Numbers

```c
int main () {
    int N, count, sum;
    scanf (“%d”, &N);
    sum = 0;
    count = 1;
    while (count <= N) {
        sum = sum + count;
        count = count + 1;
    }
    printf (“%d
”, sum);
    return 0;
}
```
Exercise

- Convert while Loop into for loop

```c
int volume = 25;
int barrelSize = 200;
while(volume < barrelSize) {
    printf("The barrel is not full.\n");
    volume = volume + 25;
}
```
Some observations on **for**

- Initialization, loop-continuation test, and update can contain arithmetic expressions
  
  ```c
  for ( k = x;   k <= 4 * x * y;   k += y / x )
  ```

- Update may be negative (decrement)
  
  ```c
  for ( digit = 9; digit >= 0; --digit)
  ```

- If loop continuation test is initially 0 (**false**)
  - Body of **for** structure not performed
    - No statement executed

- Program proceeds with statement after **for** structure
  ```c
  for (count=1; 0; ++count)  
    sum = sum + count;
  ```
Programming Exercise

- Display all even numbers from 0 to 20

```c
int i;
for(i=0;i<20;i+=2)
    printf("%d\n", i);
```
Looping: **do-while** statement

do
  
  **statement**;
  
while (expression);

Do {
  Block of
    **statements**;
} while (expression);
Example

**Problem:** Prompt user to input “month” value, keep prompting until a valid month value is given as input…

```c
do {
    printf ("Please input month {1-12}\n");
    scanf ("%d", &month);
} while ((month < 1) || (month > 12));
```
Decimal to binary conversion (prints binary in reverse order)

```c
int main() {
    int dec;
    scanf("%d", &dec);
    do{
        printf("%2d", (dec % 2));
        dec = dec / 2;
    } while (dec != 0);
    printf("\n");
    return 0;
}
```

Output

277
101010001
int main () {
    char echo ;
    do {
        scanf (“%c”, &echo);
        printf (“%c”,echo);
    } while (echo != ‘\n’) ;
    return 0;
}
Specifying “Infinite Loop”

```c
while (1) {
    statements
}
```

```c
for (; ;) {
    statements
}
```

```c
do {
    statements
} while (1);
```
The **break** Statement

- Break out of the loop body `{ }`
  - can use with `while`, `do while`, `for`, `switch`
  - does not work with `if`, `else`
- Causes immediate exit from a `while`, `do/while`, `for` or `switch` structure
- Program execution continues with the first statement after the structure
int main() {
    int fact, i;
    fact = 1;  i = 1;
    while ( i<10 ) { /* run loop –break when fact >100*/
        fact = fact * i;
        if ( fact > 100 ) {
            printf ("Factorial of %d above 100", i);
            break; /* break out of the while loop */
        }
        ++i;
    }
    return 0;
}
int main() {
    int  n, i=2;
    scanf ("%d", &n);
    limit = sqrt(n);
    for (i = 2, i <= limit; i++) {
        if (n % i == 0) {
            printf ("%d is not a prime \n", n);
            break;
        }
    }
    if (i > limit) printf ("%d is a prime \n", n);
    return 0;
}
Another Way

```c
int main() {
    int n, i = 2, flag = 0;
    double limit;
    scanf("%d", &n);
    limit = sqrt(n);
    while (i <= limit) {
        if (n % i == 0) {
            printf("%d is not a prime \n", n);
            flag = 1; break;
        }
        i = i + 1;
    }
    if (flag == 0) printf("%d is a prime \n", n);
    return 0;
}
```
The *continue* Statement

- **Skips the remaining statements in the body of a while, for or do/while structure**
  - Proceeds with the next iteration of the loop
- **while and do/while loop**
  - Loop-continuation test is evaluated immediately after the continue statement is executed
- **for loop**
  - $\text{expr3}$ is evaluated, then $\text{expr2}$ is evaluated
Example with **break** and **continue**: Add positive numbers until a 0 is typed, but ignore any negative numbers typed.

```c
int main() {
    int sum = 0, next;
    while (1) {
        scanf("%d", &next);
        if (next < 0) continue;
        else if (next == 0) break;
        sum = sum + next;
    }
    printf ("Sum = %d\n", sum) ;
    return 0;
}
```

**Output**

```
10
-20
30
40
-5
10
0
Sum = 90
```
Loops: Some Common Mistakes

while (sum <= NUM) {
    sum = sum + 2;
}

for (i = 0; i <= NUM; ++i) {
    sum = sum + i;
}

for (i = 1; i != 10; i = i + 2) {
    sum = sum + i;
}

double x;
for (x = 0.0; x != 2.0; x = x + 0.2) {
    printf("%.18f\n", x);
}
Nested Loops: Printing a 2-D Figure

How would you print the following diagram?

```
* * * * *
* * * * *
* * * * *
```

Repeat 3 times

Print a row of 5 *’s

Repeat 5 times

Printing *
Display pattern: Configurable number of rows and columns

```c
const int ROWS = 3;
const int COLS = 5;
...
row = 1;
while (row <= ROWS) {
    /* print a row of 5 *'s */
    col = 1;
    while (col <= COLS) {
        printf("* ");
        col++;
    }
    printf("\\n");
    ++row;
}
```
const int ROWS = 3;
const int COLS = 5;

....

for (row=1; row<=ROWS; ++row) {
    for (col=1; col<=COLS; ++col) {
        printf("* ");
    }
    printf("\n");
}
Another 2-D Figure

Print

```
const int ROWS = 5;
....
int row, col;
for (row=1; row<=ROWS; ++row) {
    for (col=1; col<=row; ++col) {
        printf("* ");
    }
    printf("\n");
}
```
const int ROWS = 5;

....

int row, col;
for (row=0; row<ROWS; ++row) {
    for (col=1; col<=row; ++col)
        printf("  ");
    for (col=1; col<=ROWS-row; ++col)
        printf("* ");
    printf ("\n");
}
**break and continue with nested loops**

- For nested loops, break and continue are matched with the nearest loops (for, while, do-while)

- **Example:**

```java
while (i < n) {
    for (k=1; k < m; ++k) {
        if (k % i == 0) break;
    }
    i = i + 1;
}
```

Breaks to here
```c
int main()
{
    int low, high, desired, i, flag = 0;
    scanf("%d %d %d", &low, &high, &desired);
    i = low;
    while (i < high) {
        for (j = i+1; j <= high; ++j) {
            if (j % i == desired) {
                flag = 1;
                break;       // Breaks to here
            }
        }
        if (flag == 1) break;   // Breaks to here
        i = i + 1;
    }
    return 0;
}
```
Practice Problems (do each with both for and while loops separately)

1. Read in an integer N. Then print the sum of the squares of the first N natural numbers.
2. Read in an integer N. Then read in N numbers and print their maximum and second maximum (do not use arrays even if you know it).
3. Read in an integer N. Then read in N numbers and print the number of integers between 0 and 10 (including both), between 11 and 20, and > 20. (do not use arrays even if you know it).
4. Repeat 3, but this time print the average of the numbers in each range.
5. Read in a positive integer N. If the user enters a negative integer or 0, print a message asking the user to enter the integer again. When the user enters a positive integer N finally, find the sum of the logarithmic series \( \log_e(1+x) \) up to the first N terms.
6. Read in an integer N. Then read in integers, and find the sum of the first N positive integers read. Ignore any negative integers or 0 read (so you may actually read in more than N integers, just find the sum with only the positive integers and stop when N such positive integers are read).
7. Read in characters until the ‘\n’ character is typed. Count and print the number of lowercase letters, the number of uppercase letters, and the number of digits entered.