Basis and Practice in Programming

week 4
Various Operators in C

- Substitution and arithmetic operator

<table>
<thead>
<tr>
<th>Operator</th>
<th>Example</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>a=20</td>
<td>Assign the value to variable a</td>
</tr>
<tr>
<td>+</td>
<td>a=4+3</td>
<td>Addition</td>
</tr>
<tr>
<td>-</td>
<td>a=4–3</td>
<td>Subtraction</td>
</tr>
<tr>
<td>*</td>
<td>a=4*3</td>
<td>Multiply</td>
</tr>
<tr>
<td>/</td>
<td>a=4/3</td>
<td>Division</td>
</tr>
<tr>
<td>%</td>
<td>a=4%3</td>
<td>Remainder</td>
</tr>
</tbody>
</table>
Various Operators in C

- Increment and decrement operator

<table>
<thead>
<tr>
<th>Operator</th>
<th>Example</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>++a</td>
<td>printf(&quot;%d&quot;, ++a)</td>
<td>Increase first, operate later</td>
</tr>
<tr>
<td>a++</td>
<td>printf(&quot;%d&quot;, a++)</td>
<td>Operate first, increase later</td>
</tr>
<tr>
<td>--b</td>
<td>printf(&quot;%d&quot;, --a)</td>
<td>Decrease first, operate later</td>
</tr>
<tr>
<td>b--</td>
<td>printf(&quot;%d&quot;, a--)</td>
<td>Operate first, decrease later</td>
</tr>
</tbody>
</table>
Various Operators in C

- Increment and decrement operator

```c
/* week 3 practice 1 */
#include <stdio.h>

int main(void)
{
    int val1=10;
    int val2=10;

    printf("Operate first, increase later : %d \n", val1++);
    printf("Print again : %d \n\n", val1);

    printf("Increase first, operate later : %d \n", ++val2);
    printf(" Print again : %d \n", val2);

    return 0;
}
```
Various Operators in C

- Increment and decrement operator

```c
/ * week 3 practice 2 */
#include <stdio.h>

int main(void)
{
    int val1=10;
    int val2=(val1--)+2;
    printf("val1 : %d \n", val1);
    printf("val2 : %d \n", val2);
    return 0;
}

/ * week 3 practice 3 */
#include <stdio.h>

int main(void)
{
    int val1=10;
    int val2=(-val1)+2;
    printf("val1 : %d \n", val1);
    printf("val2 : %d \n", val2);
    return 0;
}
```
Various Operators in C

- Association operators
  - Represents association between two operands
  - Returns true(1) or false(0)

<table>
<thead>
<tr>
<th>Operator</th>
<th>Example</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>a&lt;b</td>
<td>Is a less than b?</td>
</tr>
<tr>
<td>&gt;</td>
<td>a&gt;b</td>
<td>Is a larger than b?</td>
</tr>
<tr>
<td>==</td>
<td>a==b</td>
<td>Is a equal to b?</td>
</tr>
<tr>
<td>!=</td>
<td>a!=b</td>
<td>Is a not equal to b?</td>
</tr>
<tr>
<td>&lt;=</td>
<td>a&lt;=b</td>
<td>Is a less than or equal to b?</td>
</tr>
<tr>
<td>&gt;=</td>
<td>a&gt;=b</td>
<td>Is a larger than or equal to b?</td>
</tr>
</tbody>
</table>
Various Operators in C

• Association operators

```c
/* week 3 practice 4 */
#include <stdio.h>

int main(void)
{
    int val1=10;
    int val2=12;
    int result1, result2, result3;

    result1=(val1==val2);
    result2=(val1<=val2);
    result3=(val1>val2);

    printf("result1 : %d \n", result1);
    printf("result2 : %d \n", result2);
    printf("result3 : %d \n", result3);

    return 0;
}
```
Various Operators in C

- Logic operators
  - Represents and, or, not
  - Returns true(1) or false(0)

<table>
<thead>
<tr>
<th>Operator</th>
<th>Example</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;&amp;</td>
<td>a&amp;&amp;b</td>
<td>Returns true(1) when both a and b are true</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>!</td>
<td>!a</td>
<td>Returns false when a is true and vice versa</td>
</tr>
</tbody>
</table>
Various Operators in C

- Logic operators

```c
/* week 3 practice 5 */
#include <stdio.h>

int main(void)
{
    int val1=10;
    int val2=12;
    int result1, result2, result3;

    result1=(val1==10 && val2==12);
    result2=(val1<12 || val2>12);
    result3=(!val1);

    printf("result1 : %d \n", result1);
    printf("result2 : %d \n", result2);
    printf("result3 : %d \n", result3);

    return 0;
}
```
Various Operators in C

- Bit manipulation operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Example</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&amp;</code></td>
<td>a&amp;b</td>
<td>Binary AND Operator copies a bit to the result if it exists in both operands.</td>
</tr>
<tr>
<td>`</td>
<td>`</td>
<td>a</td>
</tr>
<tr>
<td><code>^</code></td>
<td>a^b</td>
<td>Binary XOR Operator copies the bit if it is set in one operand but not both.</td>
</tr>
<tr>
<td><code>~</code></td>
<td>~a</td>
<td>Binary Ones Complement Operator is unary and has the effect of 'flipping' bits.</td>
</tr>
<tr>
<td><code>&lt;&lt;</code></td>
<td>a&lt;&lt;2</td>
<td>Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.</td>
</tr>
<tr>
<td><code>&gt;&gt;</code></td>
<td>a&gt;&gt;2</td>
<td>Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.</td>
</tr>
</tbody>
</table>
Various Operators in C

- Bit manipulation operators
  - Example: and operator

<table>
<thead>
<tr>
<th>Operand A</th>
<th>Operand B</th>
<th>Result</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>

A : 0000 0101
B : 0000 0100

Result: 0000 0100
Various Operators in C

- Bit manipulation operators
  - Example: or operator

<table>
<thead>
<tr>
<th>Operand A</th>
<th>Operand B</th>
<th>Result</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>

A: 0000 0101
B: 0000 0100
Result: 0000 0101
Various Operators in C

- Bit manipulation operators
  - Example: xor operator

<table>
<thead>
<tr>
<th>Operand A</th>
<th>Operand B</th>
<th>Result</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>

A : 0000 0101
B : 0000 0100

Result: 0000 0001
Various Operators in C

- Bit manipulation operators
  - Example: binary one’s complement operator

<table>
<thead>
<tr>
<th>Operand A</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

A : 0000 0101
----------------
Result: 1111 1010
Various Operators in C

- Bit manipulation operators
  - Example: shift operator

```c
int a = 5;
a = a << 1;
```

```
A : 0000 0101
---------------
Result: 0000 1010
```
If Statement (1/2)

• If-else statement
  – Control flow statement
  – Controlled by given condition

```c
if (a == 1)
    printf("a is 1");
else
    printf("a isn’t 1");
```

- If `a == 1` is true, `printf("a is 1");` is executed.
- If `a == 1` is false, `printf("a isn’t 1");` is executed.
If Statement (2/2)

• If statement (continued)

```c
/* practice 5 : if statement exercise*/
#include <stdio.h>

int main(void)
{
    int a;
    scanf("%d", &a);

    if (a > 10) {
        printf("a is bigger than 10\n"); // condition is true
    } else {
        printf("a is smaller than 10\n"); // condition is false
    }

    return 0;
}
```
Exercise 1

• Letter changer
  – Enter one character (only Alphabet can be input)
  – Translate small letter to capital letter
  – Translate capital letter to small letter
  – Print translated letter as output
  – Must use if-statement

```
a   A
Z   z
```