CENG 230

*Introduction to C Programming*

Week 6 – Repetition

Sinan Kalkan

Some slides/content are borrowed from Tansel Dokeroglu, Nihan Kesim Cicekli.
C accepts any nonzero value as a representation of true.
short-circuit evaluation

An expression of the form \((a \text{ || } b)\) must be true if \(a\) is true. Consequently, C stops evaluating the expression when it determines that the value of \(!\text{flag}\) is 1 (true).

Similarly, an expression of the form \((a \text{ && } b)\) must be false if \(a\) is false, so C would stop evaluating such an expression if its first operand evaluates to 0.
Writing English Conditions in C

- **x** is 3.0
- **y** is 4.0
- **z** is 2.0

<table>
<thead>
<tr>
<th>English Condition</th>
<th>Logical Expression</th>
<th>Evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>x and y are greater than z</td>
<td>x &gt; z &amp;&amp; y &gt; z</td>
<td>1 &amp;&amp; 1 is 1 (true)</td>
</tr>
<tr>
<td><strong>x</strong> is equal to 1.0 or 3.0</td>
<td>x == 1.0</td>
<td></td>
</tr>
<tr>
<td><strong>x</strong> is in the range z to y, inclusive</td>
<td>z &lt;= x &amp;&amp; x &lt;= y</td>
<td>1 &amp;&amp; 1 is 1 (true)</td>
</tr>
<tr>
<td><strong>x</strong> is outside the range z to y</td>
<td>!(z &lt;= x &amp;&amp; x &lt;= y)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>!x &lt;= y</td>
<td>0</td>
</tr>
</tbody>
</table>
Comparing Characters

<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>'B' &lt;= 'A'</td>
<td>0 (false)</td>
</tr>
<tr>
<td>'Z' == 'z'</td>
<td>0 (false)</td>
</tr>
<tr>
<td>'a' &lt;= ch &amp;&amp; ch &lt;= 'z'</td>
<td>1 (true) if <code>ch</code> is a lowercase letter</td>
</tr>
</tbody>
</table>
Examples

int a = 6, b = 9, c = 14, flag = 1.

c == a + b || !flag
a != 7 && flag || c >= 6
!(b <= 12) && a % 2 == 0
!(a > 5 || c < a + b)

int ans;
int p = 100, q = 50.

ans = (p > 95) + (q < 95);
What is the value of ans?
Complement the expression below

a != 7 && flag || c >= 6
a == 7 || flag && c < 6

!(1 || 0) 0
!(1 || 1 && 0) 0
!( (1 || 0) && 0 ) 1 (Parenthesis are useful)
if (x > 0.0)
    pos_prod = pos_prod * x;

if (crsr_or_frgt == 'C')
    printf("Cruiser\n");
else
    printf("Frigate\n");

It displays either Cruiser or Frigate, depending on the character stored in the type char variable crsr_or_frgt.

if crsr_or_frgt == 'C'  /* error - missing parentheses */
    printf("Cruiser\n");
    printf("Combat ship\n");

if (crsr_or_frgt == 'C');  /* error - improper placement of ; */
    printf("Cruiser\n");
    printf("Combat ship\n");
nested if statements and alternative decisions

if statement inside another

```python
if (x > 0)
    num_pos = num_pos + 1;
else
    if (x < 0)
        num_neg = num_neg + 1;
    else /* x equals 0 */
        num_zero = num_zero + 1;
```
Multiple-Alternative Decision Form of Nested if

SYNTAX:

```c
if (condition_1)
    statement_1
else if (condition_2)
    statement_2
else if (condition_n)
    statement_n
else
    statement_e
```

EXAMPLE:

```c
/* increment num_pos, num_neg, or num_zero depending on x */
if (x > 0)
    num_pos = num_pos + 1;
else if (x < 0)
    num_neg = num_neg + 1;
else /* x equals 0 */
    num_zero = num_zero + 1;
```
<table>
<thead>
<tr>
<th>Loudness in Decibels (db)</th>
<th>Perception</th>
</tr>
</thead>
<tbody>
<tr>
<td>50 or lower</td>
<td>quiet</td>
</tr>
<tr>
<td>51 – 70</td>
<td>intrusive</td>
</tr>
<tr>
<td>71 – 90</td>
<td>annoying</td>
</tr>
<tr>
<td>91 – 110</td>
<td>very annoying</td>
</tr>
<tr>
<td>above 110</td>
<td>uncomfortable</td>
</tr>
</tbody>
</table>

```c
/* Display perception of noise loudness */
if (noise_db <= 50)
    printf("%d-decibel noise is quiet.\n", noise_db);
else if (noise_db <= 70)
    printf("%d-decibel noise is intrusive.\n", noise_db);
else if (noise_db <= 90)
    printf("%d-decibel noise is annoying.\n", noise_db);
else if (noise_db <= 110)
    printf("%d-decibel noise is very annoying.\n", noise_db);
else
    printf("%d-decibel noise is uncomfortable.\n", noise_db);
```
/* incorrect perception of noise loudness */

if (noise_db <= 110)
        printf("%d-decibel noise is very annoying.\n", noise_db);
else if (noise_db <= 90)
        printf("%d-decibel noise is annoying.\n", noise_db);
else if (noise_db <= 70)
        printf("%d-decibel noise is intrusive.\n", noise_db);
else if (noise_db <= 50)
        printf("%d-decibel noise is quiet.\n", noise_db);
else
        printf("%d-decibel noise is uncomfortable.\n", noise_db);
Switching values of two variables

```c
if (x > y) {
    temp = x;
    x = y;
    y = temp;
}

/* Switch x and y */
/* Store old x in temp */
/* Store old y in x */
/* Store old x in y */
```

<table>
<thead>
<tr>
<th>Statement Part</th>
<th>x</th>
<th>y</th>
<th>temp</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>if (x &gt; y) {</td>
<td>12.5</td>
<td>5.0</td>
<td>?</td>
<td>12.5 &gt; 5.0 is true.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Store old x in temp.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>12.5</td>
<td>Store old y in x.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5.0</td>
<td></td>
<td>Store old x in y.</td>
</tr>
<tr>
<td></td>
<td>12.5</td>
<td></td>
<td>12.5</td>
<td></td>
</tr>
</tbody>
</table>
17- What is the output of the following program segment?

```
int x = -1;
if (x++==0) printf("%d\n",x);
else if(++x>1) printf("%d",x);
else printf("%d",x);
```

a) -1  

b) 0  

c) 1  

d) 2  

e) 3

---

18. For what exact range of values of variables a and b, does the following code segment display the value 0?

```
m = -1;
if (a>20)
    if (b<10)
        if (a>=30)
            m = 4;
        else
            m = 0;
    else
        m = 1;
else
    m = 2;
printf("%d",m);
```

a) a > 20
   b ≥ 10

b) 20 ≤ a ≤ 30
   b ≤ 10

c) 20 < a < 30
   b < 10

d) a ≥ 30
   b < 10

e) 20 < a < 30
   b ≥ 10
19- Assuming that x, y and flag are integers, what is the value printed by the following if statements?

```
if(x>y)
    if(x>z) printf("%d", x);
else
    if(z>y) printf("%d",z);
else
    if(y>z) printf("%d",y);
else printf("%d",z);
```

a) minimum  b) maximum  c) median
d) last      e) indeterminate

20- What is the output of the following program segment?

```
int x=6, y=3, A=3, B=5, C=7;
if (x<A && y>B)
    if (y>0)
        printf("A");
    else printf("B");
    else if (y>C || x>0)
        printf("C");
```

a) A        b) B        c) C        d) AC       e) no output
27) What will be the output of the program?
#include<stdio.h>
void main()
{
    int a = 4;
    if(a == 4)
        printf("a1");
    else
        printf("a2");
        printf("a3");
    printf("a4"); }

a) a4       b) a1a4       c) a2a3a4       d) a1a3a4       e) a2a3

28) What will be the output of the program?
#include<stdio.h>
void main()
{
    int a = 9, b = 3;
    if(!a <= 4 )
        b = 5;
        a = 1;
    printf("a=%d b=%d\n", a, b); }

a) a = 9, b = 3
b) a = 4, b = 3
c) a = 1, b = 5
d) a = 9, b = 5
e) a = 1, b = 3
32) What will be the output of the program?

```c
#include<stdio.h>
void main()

    int m=8;
    float n=8.6;
    if (m > n)
        
    else {
        m = n * 2;
        n = n / 2;
        
        printf("%d %f ", m, n);
    }

a) 17 4.300000
b) 17 4.000000
c) 16 4.300000
d) 16 4.000000
e) Compile error
```

31) What will be the output of the program?

```c
#include<stdio.h>
void main()

    int z=9;
    z=z-4;
    if (z<9 || ++z>4) z=z+2;
    printf("%d ", z);
```

a) 5 b) 6 c) 7 d) 8 e) 9
Changing the flow of the program

- Multi-way conditionals: switch statements

```c
switch(expr)
{
    case value-1:
        ....
        break;
    case value-2:
        ....
        break;
    default:
        ....
        break;
}
```
Example

• main()
  {
    int i=3;
    switch(i)
    {
      default: printf("zero");
      case 1: printf("one");
          break;
      case 2: printf("two");
          break;
      case 3: printf("three");
          break;
    }
  }

• main()
  {
    int i=1;
    switch(i)
    {
      default: printf("zero");
      case 1: printf("one");
      case 2: printf("two");
          break;
      case 3: printf("three");
          break;
    }
  }
Simple Macros

• For long and/or frequent constants:
  • \texttt{#define PI 3.14159265}

• For long and/or frequent calculations:
  • \texttt{#define Area(Radius) (4*PI*Radius*Radius)}
  • \texttt{... a = 10.0 + Area(2.0);}
Remember Types of Errors?

Compile time → Syntax errors
- When the code violated the grammar rules of C.
- Compiler detects these errors

Run-time errors
- Happen when the program directs the computer to an illegal operation.
- Such as Division by zero

Logic errors
- A faulty algorithm
- It gives no error message.
Run-time error (division by zero)

If the value of variable “second” is given as zero
Logic error

```c
#include <stdio.h>

int main(void)
{
    int first, second, ans;

    printf("Enter two integers> ");
    scanf("%d%d", &first, &second);

    ans = first * second;
    printf("The sum of the number is : %d\n", ans);

    system("pause");
    return (0);
}
```
Sample questions

Evaluate the following expressions with 7 and 22 as operands. (be careful that the values are integers)

$$\frac{22}{7} \quad \frac{7}{22} \quad 22 \mod 7 \quad 7 \mod 22$$

Evaluate the following, assuming that letters have consecutive character codes.

a. $(\text{int})'D' - (\text{int})'A'$
b. $(\text{char})(\text{int})'C' + 2)$
c. $(\text{int})'6' - (\text{int})'7'$
4) What will be the output of the following C program?

```c
#include <stdio.h>
define X 5+3
int main() {
    int a = X / 2;
    printf("%d", a);
    return 0;
}
```

a) 4  b) 6  c) 8  d) 6.5

5) What will be the output of the following C program?

```c
#include <stdio.h>
int main() {
    double x, y;
    x = 7;
    x = x / 2;
    y = x + x / 2;
    printf("%.2f %.2f", x, y);
    return 0;
}
```

a) 3.00 3.00  b) 3.0 4.50  c) 3.50 3.50  d) 3.50 5.25  e) 4.50 5.25

6) What will be the output of the following C program?

```c
#include <stdio.h>
int main
    (void) {
        int a; double b; printf("%d %.2f", a=5+3/2, b=5+3/2);
        return 0;
    }
```

a) 6 6.00  b) 6 6.50  c) 7 7.00  d) 7 7.50  e) This program will not compile successfully because of bad indentation.
# include <stdio.h>
int main (void) {
    printf ("%c,%d,%c,%d", 'a', 'a', 97, 97);
    return 0;
}

a) a,97,a,98       b) 97,a,97,a   c) 97,97,a,a   d) a,97,a,97   e) a,97,97,a

9) What will be the output of the following code segment?
   double pi = 22/7;
   printf("%3.2f", pi);
   a)3.142857   b)3.14   c)3.14   d)03.14   e)3.00

10) What would be the output after execution of the following code?
    int x=5, y=3;
    y+=5-y+x++;  
    x=y%y;
    printf("%d", x);
    a) 3       b) 5       c) 6       d) 4       e) 2
12) What would be the output after execution of the following code?

    int x=2;
double y=22/5*(double)x;
printf("%.2f", y);

a) 2.20  b) 2.00  c) 8.00  d) 8.80  e) 2.80

16) What would be the output after execution of the following code?

    int b,a=3,c=5;
b=12+a--/++c-(-a);
printf("%d", b);

a) 10  b) 12  c) 9  d) 11  e) 8

6) What could be the output of the following code segment?

printf("%07.4f", 22/7.0);

a) 03.143
b) 003.143
c) .314285
d) 03.1429
 e) .003143
15) What will be the output of the following code segment?
```c
int a=4, b=3;
a = 4*3-2+b-/-2*3%2*4-2;
printf("%d", a--);
```

- **a)** 8
- **b)** 7
- **c)** 12
- **d)** 10
- **e)** 11

Hint: `b--/2*3%2*4` : execute this part from left to right

2) What will be the output after the input of 13?
```c
int s;
scanf("%d", &s);
printf("%d", s%2-+-s);
```

- **a)** 12
- **b)** 6
- **c)** 7
- **d)** 8
- **e)** None of them

4) What will be the output after the input of 7?
```c
int x;
printf("Enter a number");
scanf("%d", &x);
printf("%d %d %d", x - 1, x, x--);
```

- **a)** 5 6 7
- **b)** 4 6 6
- **c)** 6 7 6
- **d)** 6 7 7
- **e)** 4 5 6
4) If $a$ is 5, $b$ is 4, $c$ is 10 what is the output?

```c
a=b=c+6%2;
printf("%d %d %d", a, b, c);
```

a) 5 10 10  b) 13 13 10  c) 10 10 10  d) 5  4  10  e) 13 10 10

5) What is the output of the below code segment?

```c
int i=32;
char c;
c=i;
printf("%d", c);
```

a) 23  b) 'c'  c) 69  d) 'E'  e) 32

8) What is the C equivalent of the following expression?

\[ x = \frac{-b - \sqrt{b^2 - 4ac}}{2a} \]

a) \(x=(\text{-}b\text{-}\text{sqrt}(b\text{*}b\text{-}4\text{*a}\text{*c}))/(2\text{*a})\)

b) \(x=(\text{-}b\text{-}\text{sqrt}(b\text{*}2\text{-}4\text{*a}\text{*c}))/2\text{*a}\)

c) \(x=(\text{-}b\text{-}\text{sqrt}(b\text{*}2\text{-}4ac))/2a\)

d) \(x=((\text{-}b)\text{-}\text{sqrt}(\text{b}\text{*}2\text{-}4ac))/2a\)

e) \(x=\text{-}b\text{-}\text{sqrt}(b\text{*}b\text{-}4\text{*a}\text{*c})/2\text{*a}\)
Today

• Conditional Expressions
• Repetitions
  • Iterative statements
    • while
    • do-while
    • for
Conditional Expression Operator

\[ y = x > 3 \ ? \ a+1 : a-1; \] means

\begin{align*}
\text{if} \ (x > 3) \\
y &= a+1; \\
\text{else} \\
y &= a-1;
\end{align*}

\[ z = (a > b) \ ? \ a : b; \] \ (finds maximum)

\[ \text{Printf} \("%d%c", k, (k\%10 == 9) \ ? \ ‘A’ : ‘a’); \]
Loops, iterations, repetitions

while, do-while and for statements
Most programs involve repetition or looping.

A loop is a group of instructions the computer executes repeatedly while some loop-continuation condition remains true.

```c
/* Fig. 4.1: fig04_01.c */
#include <stdio.h>

/* function main begins program execution */
int main( void )
{
    int counter = 1; /* initialization */

    while ( counter <= 10 ) { /* repetition condition */
        printf ( "%d\n", counter ); /* display counter */
        ++counter; /* increment */
    } /* end while */

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

• while loop

**Initialization**;

while( expr )

statement;


**Initialization**;

while( expr )

{
    statement;
    statement;
    statement;

}

• Bad examples:

while( x = 1)
{
    x = getchar();
}

while( x != 1.0 )
{
    x += 0.005;
}

x = 0.0;
while( x != 1.0 )
{
    x += 0.005;
}
Example

• Factorial

```c
int N, fact = 1;
scanf("%d", &N);
while( N > 0 )
{   fact *= N--;
}```
Repetitions

• do-while loop
  
  *Initialization*
  
  do
  
  statement
  
  while( `expr` );
  
  statement;

  do
  {
    x = getchar();
    putchar(x);
  }

  while( x != EOF );
Example

• Factorial with do-while:

```c
int N, fact = 1;
scanf("%d", &N);
do
{ fact *= N--; }
while( N > 0 );
```
Finding power of a number

```c
/* C program to calculate the power of an integer*/
#include <stdio.h>
int main()
{
    int base, exp;
    long long int value=1;
    printf("Enter base number and exponent respectively: ");
    scanf("%d%d", &base, &exp);
    while (exp!=0)
    {
        value*=base; /* value = value*base; */
        --exp;
    }
    printf("Answer = %d\n", value);
    system("pause");
}
```