CS101: Problem Solving through C Programming

Sachchida Nand Chaurasia

Assistant Professor

Department of Computer Science
Banaras Hindu University
Varanasi
Email id: snchaurasia@bhu.ac.in, sachchidanand.mca07@gmail.com

January 29, 2021

Conditional statements I

```
4 + 2 * 3 > 12 - 2
ANSWER: ??

! (3 > 4)
ANSWER: ??

!((-5 >= -6.2) || (7 != 3) && (6 == (3 + 3)))
 ANSWER: ??


4 % 2 == 0 <= 8 * 2
 ANSWER: ??

10 % 4 * 3 - 8 <= 18 + 30 / 4 - 20
ANSWER: ??

month is 2, year is 1999, (month == 2) && ((year % 4) == 0)
ANSWER: ??

!( 1 || 0 )
ANSWER: ??

!( 1 || 1 && 0 )
 ANSWER: ??

!( ( 1 || 0 ) && 0 )
```

Conditional statements II

```
27 ANSWER: ??
28
29 8 & 1 == 1000 & 0001 == 0000
30 ANSWER: ??
31
32 7 & 1 == 0111 & 0001 == 0001
33 ANSWER: ??
34
35
```

Control statements in C I

Control statements are used to alter the flow of the program. They are used to specify the order in which statements can be executed. They are commonly used to define how control is transferred from one part of the program to another. C language has following control statements:

- if. . . else
- switch
- Loops
  - for
  - while
  - do... while

Compound statement I

A Compound statement is a block of statement grouped together using braces (). In a compound statement, all statements are executed sequentially. A compound statement is also known as a block. It takes the following form:

```
{
        statement1;
        statement2;
        statement3;
        ...
        statementn;
}
Output: ???

```

Compound statement II

```
#include<stdio.h>
int main()
{
	int i = 100;
	printf("A single statement\n");
	{
		// a compound statement
		printf("A statement inside compound statement\n");
		printf("Another statement inside compound statement\n");
	}
	// signal to operating system everything works fine
	return 0;
}
Expected Output:
A single statement
A statement inside compound statement
Another statement inside compound statement
Output: ???

```

if statement I

If statement is used to test a condition and take one of the two possible actions.

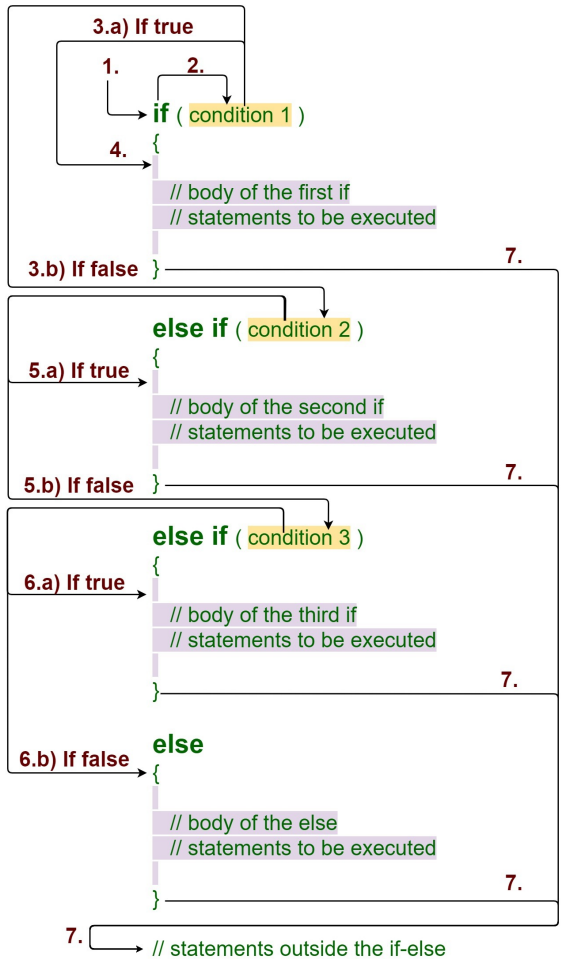
The syntax of the if statement is:

```
if (condition)
{
       // if block
       statement1;
       statement2;
}

```

if statement II

If - else if ladder



if statement III

The *condition* can be any constant, variable, expression, relational expression, logical expression and so on. Just remember that in C, any non-zero value is considered as true while 0 is considered as false.

```
#include<stdio.h>
int main()
{
	int n;
	printf("Enter a number: ");
	scanf("%d", &n);
	if(n % 2 == 0)
	{
		printf("%d is even", n);
	}
	// signal to operating system everything works fine
	return 0;
}
Expected Output:
Enter a number: 46
46 is even
Enter a number: 21
Output: ???

```

if statement IV

Which statement belongs to if?:

```
1 if (condition)
2 statement1;
3 statement2;
4 statement3;
5 Output: ???
6
```

```
1  #include<stdio.h>
2
3  int main()
4  {
5        if(0)
6        printf("statement 1\n");
7        printf("statement 2\n");
8        printf("statement 3\n");
9
10       // signal to operating system everything works fine
11       return 0;
12 }
13 Output: ???
14
```

if statement V

The else clause: The else clause allows us to add an alternative path to the if condition. Statements under the else block are executed only when the if condition is false.

```
if (condition)
{
	// if block
	statement1;
	statement2;
}

else
{
	// else block
	statement3;
	statement4;
}
Output: ???

```

if statement VI

```c
#include<stdio.h>

int main()
{
	int n;
	printf("Enter a number: ");
	scanf("%d", &n);
	if(n % 2 == 0)
	{
		printf("%d is even", n);
	}
	else
	{
		printf("%d is odd", n);
	}
	// signal to operating system everything program ran fine
	return 0;
}
Output: ???

```

if statement VII

Nesting if. . . else: We can add if.. else statement inside if block or else block. This is called nesting of if.. else.

```
if(condition1)
{
	if(condition2)
	{
		statement1;
		statement2;
	}
	else
	{
		statement3;
		statement4;
	}
}
else
{
	if(condition3)
	{
		statement5;
		statement6;
	}
	else
	{
```

if statement VIII

```
23              statement7;
24              statement8;
25      }
26  }
27  Output: ???
28
```

✓ We can nest if.. else statement to any depth.

```
1   #include<stdio.h>
2   int main()
3   {
4       int a, b, c, larger;
5       printf("Enter three numbers: ");
6       scanf("%d %d %d", &a, &b, &c);
7       if(a > b)
8       {
9           if(a > c)
10          {
11              larger = a;
12          }
13          else
14          {
15              larger = c;
16          }
```

if statement IX

```
17        }
18        else
19        {
20              if(b > c)
21              {
22                    larger = b;
23              }
24              else
25              {
26                    larger = c;
27              }
28        }
29        printf("Largest number is %d", larger);
30        // signal to operating system everything works fine
31        return 0;
32  }
33  Output: ???
34
```

if statement X

Matching if.. else parts: Sometimes it becomes confusing to associate an else clause with the if statement. Consider the following example:

```
1 if(a<10)
2   if (a % 2 ==0)
3   printf("a is even and less than 10\n");
4 else
5  printf("a is greater than 10");
6 Output: ???
7
```

✓ We can always avoid such complications using braces ().

if statement XI

```
1  if(a < 10)
2  {
3         if (a % 2 ==0)
4         {
5                printf("a is even and less than 10\n");
6         }
7
8         else
9         {
10               printf("a is greater than 10");
11        }
12 }
13 Output: ???
14
```

if statement XII

else if clause: if-else is a bi-directional statement that is used to test a condition and take one of the possible two actions.

```
1  if(condition1)
2  {
3        statement1;
4  }
5  else if(condition2)
6  {
7        statement2;
8  }
9  else if(condition3)
10 {
11       statement3;
12 }
13 ...
14 else
15 {
16       statement4;
17 }
18 Output: ???
19
```

if statement XIII

✓ Using else-if clause we can write nested if-else statement in a more compact form.

```c
#include<stdio.h>
int main()
{
	int a, b, c, larger;
	printf("Enter three numbers: ");
	scanf("%d %d %d", &a, &b, &c);
	if(a > b && a > c)
	{
		larger = a;
	}
	else if(b > a && b > c)
	{
		larger = b;
	}
	else
	{
		larger = c;
	}
	printf("Largest number is %d", larger);
	// signal to operating system everything works fine
	return 0;
}
```

if statement XIV

```
23 Output: ???
24
```

if statement XV

Ask question:

```
#include<stdio.h>

int main()
{
        int a=12;
        if(0 && ++a)
        printf("statement 1\n");
        printf("statement 2\n");
        printf("statement 3\n");

        printf("a=%d\n",a);

        // signal to operating system everything works fine
        return 0;
}
Output: ???

```

if statement XVI

```c
// C program to demonstrate working of logical operators
#include <stdio.h>

int main()
{
	int a = 10, b = 4, c = 10, d = 20;

	// logical operators

	// logical AND example
	if (a > b && c == d)
	printf("a is greater than b AND c is equal to d\n");
	else
	printf("AND condition not satisfied\n");

	// logical AND example
	if (a > b || c == d)
	printf("a is greater than b OR c is equal to d\n");
	else
	printf("Neither a is greater than b nor c is equal "
	" to d\n");

	// logical NOT example
	if (!a)
	printf("a is zero\n");
```

if statement XVII

```
26        else
27        printf("a is not zero");
28
29        return 0;
30  }
31  Output: ???
32
```

```
1   #include <stdbool.h>
2   #include <stdio.h>
3   int main()
4   {
5         int a = 10, b = 4;
6         bool res = ((a == b) && printf("GeeksQuiz"));
7         return 0;
8   }
9   Output: ???
10
```

if statement XVIII

```
#include <stdbool.h>
#include <stdio.h>
int main()
{
       int a = 10, b = 4;
       bool res = ((a != b) && printf("GeeksQuiz"));
       return 0;
}
Output: ???

```

```
#include <stdbool.h>
#include <stdio.h>
int main()
{
       int a = 10, b = 4;
       bool res = ((a != b) || printf("GeeksQuiz"));
       return 0;
}
Output: ???

```

if statement XIX

```c
#include <stdbool.h>
#include <stdio.h>
int main()
{
       int a = 10, b = 4;
       bool res = ((a == b) || printf("GeeksQuiz"));
       return 0;
}
Output: ???

```

```c
 /* C Relational Operations on integers */
#include <stdio.h>
int main()
{
         int a = 9;
         int b = 4;
         printf(" a >  b: %d \n", a > b);
         printf("a >= b: %d \n", a >= b);
         printf("a <= b: %d \n", a <= b);
         printf("a <  b: %d \n", a < b);
         printf("a == b: %d \n", a == b);
         printf("a != b: %d \n", a != b);
}
```

if statement XX

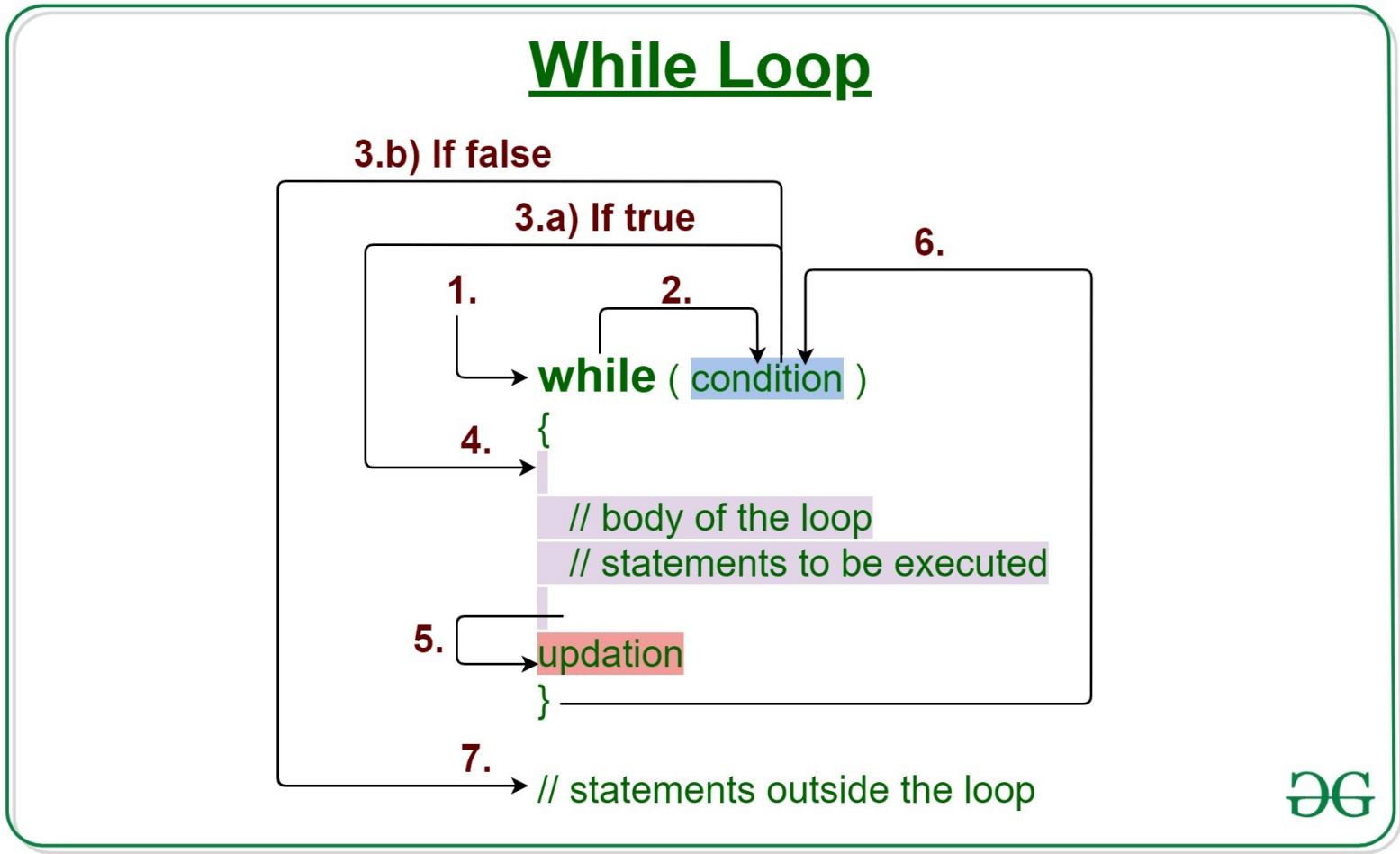
```c
/* Using C Relational Operators in If Condition */
#include <stdio.h>
void main()
{
        int x = 10;
        int y = 25;
        if (x = y)
      {
            printf(" x is equal to y \n" );
      }
        else
      {
            printf(" x is not equal to y \n" );
      }
}
```

The while loop I

Loops are used to execute statements or block of statements repeatedly. For example, suppose we want to write a program to print "Hello" 10 times. One way to achieve this is to write the following statement 10 times.

```
while(condition)
{
	// body of while loop
	statement 1;
	statement 2;
}
```

The while loop II



The while loop III

✓ Just like the if-else statement, the while loop starts with a condition.

✓ First, the condition is evaluated, if it is true then the statements in the body of the while are executed.

✓ After executing the body of the while loop, the condition is checked again, if it is still true then once again statements in the body of the while are executed.

✓ This process keeps repeating until the condition becomes false.

Properties of while loop:

- A conditional expression is used to check the condition.
- The statements defined inside the while loop will repeatedly execute until the given condition fails.

The while loop IV

- The condition will be true if it returns 0. The condition will be false if it returns any non-zero number.
- In while loop, the condition expression is compulsory.
- Running a while loop without a body is possible.
- We can have more than one conditional expression in while loop.
- If the loop body contains only one statement, then the braces are optional.

The while loop V

```c
// Print numbers from 1 to 5

#include <stdio.h>
int main()
{
	int i = 1;

	while (i <= 5)
	{
		printf("%d\n", i);
		++i;
	}
	return 0;
}
```

The while loop VI

```c
#include<stdio.h>
int main()
{
	int i = 1;
	// keep looping while i < 100
	while(i < 100)
	{
		// if i is even
		if(i % 2 == 0)
		{
			printf("%d ", i);
		}
		i++; // increment i by 1
	}
	return 0;
}
```

The while loop VII

✓ The following program calculates the sum of digits of a number entered by the user.

```
1  int main()
2  {
3        int n, num, sum = 0, remainder;
4        printf("Enter a number: ");
5        scanf("%d", &n);
6        num = n;
7        // keep looping while n > 0
8        while( n > 0 )
9        {
10             remainder = n % 10; // get the last digit of n
11             sum += remainder; // add the remainder to the sum
12             n /= 10;  // remove the last digit from n
13       }
14       printf("Sum of digits of %d is %d", num, sum);
15       // signal to operating system everything works fine
16       return 0;
17 }
```

The while loop VIII

```c
#include<stdio.h>
void main ()
{
	int j = 1;
	while(j+=2,j<=10)
	{
		printf("%d ",j);
	}
	printf("%d",j);
}
```

```c
#include<stdio.h>
void main ()
{
	while()
	{
		printf("hello Javatpoint");
	}
}
```

The while loop IX

✓ Infinitive while loop in C.

```c
while(1)
{
	//statement
}
```

```c
#include<stdio.h>
void main ()
{
	int x = 10, y = 2;
	while(x+y-1)
	{
		printf("%d %d",x--,y--);
	}
}
```

The while loop X

```c
#include <stdio.h>
int main()
{
	int i=1, j=1;
	while (i <= 4 || j <= 3)
	{
		printf("%d %d\n",i, j);
		i++;
		j++;
	}
	return 0;
}
```

The while loop XI

```c
#include <stdio.h>
int main()
{
	int i=1;
	short j=32766;
	while (i <= 4 || j >= 3)
	{
		printf("%d %d\n",i, j);
		i++;
		j++;
	}
	return 0;
}
Output: ???
```

```c
int i = 1;
while(i<10)
{
	printf("%d\n", i);
}
```

The while loop XII

```
int i = 1;
while(i=10)
{
	printf("%d", i);
}
```

```
float f = 2;
while(f != 31.0)
{
	printf("%f\n", f);
	f += 0.1;
}
```

✓ This loop is infinite because computers represent floating point numbers as approximate numbers, so 3.0 may be stored as 2.999999 or 3.00001. So the condition (f != 31.0) never becomes false. To fix this problem write the condition as f <= 31.0.

The while loop XIII

```
1 int i = 0;
2
3 while(i<=5);
4 printf("%d\n", i);
```

```
1 int i = 0;
2 while(i<=5)
3 {
4       ; // a null statement
5 }
6
7 printf("%d\n", i);
```