Programming with C and C++

CSC-101 (Lecture 10)

Dr. R. Balasubramanian
Professor
Department of Computer Science and Engineering
Mehta Family School of Data Science and Artificial Intelligence
Indian Institute of Technology Roorkee
Roorkee 247 667

bala@cs.iitr.ac.in
https://faculty.iitr.ac.in/cs/bala/

Shift Operator

</> source code

```c
#include <stdio.h>

int main(void) {
    // your code goes here
    int a=60;
    int b=2;
    int c=a<<b;
    printf("%d",c);
    return 0;
}
```

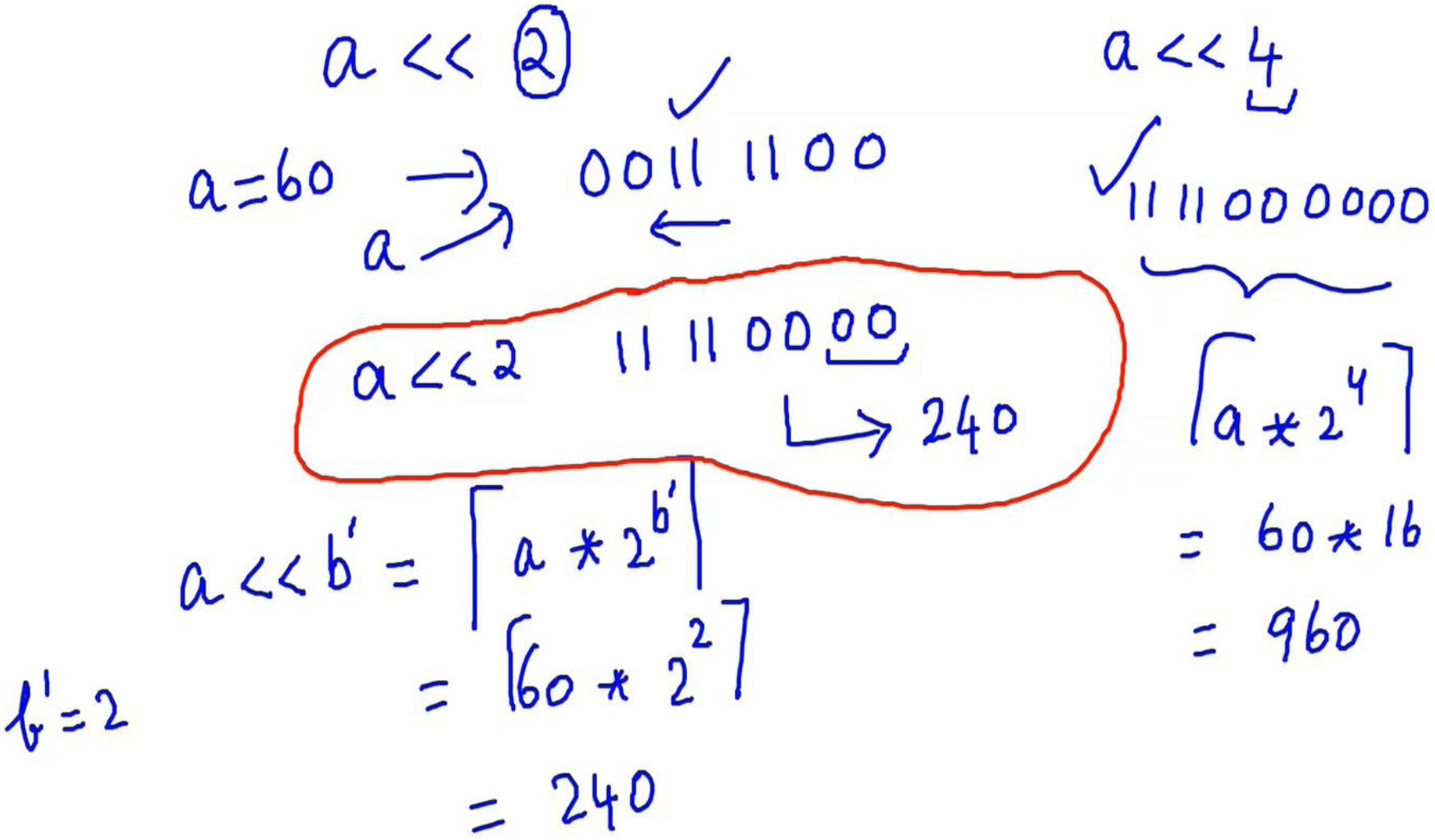
input Output

Success #stdin #stdout 0s 5544KB

240

https://ideone.com/rv4fP2

Shift Operator



Shift Operator

</> source code

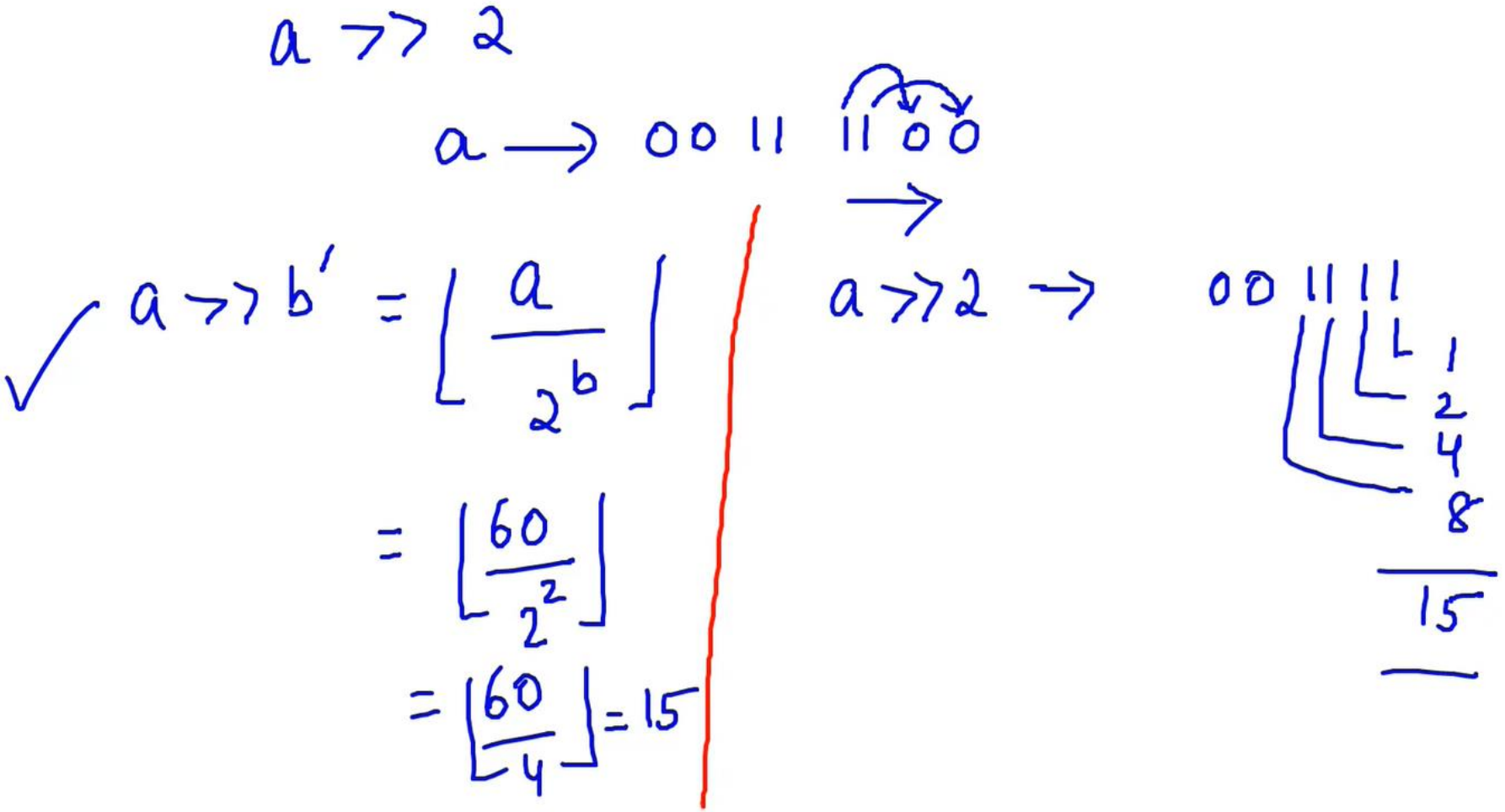
```c
#include <stdio.h>

int main(void) {
    // your code goes here
    int a=60;
    int b=2;
    int c=a>>b;
    printf("%d",c);
    return 0;
}
```

stdout

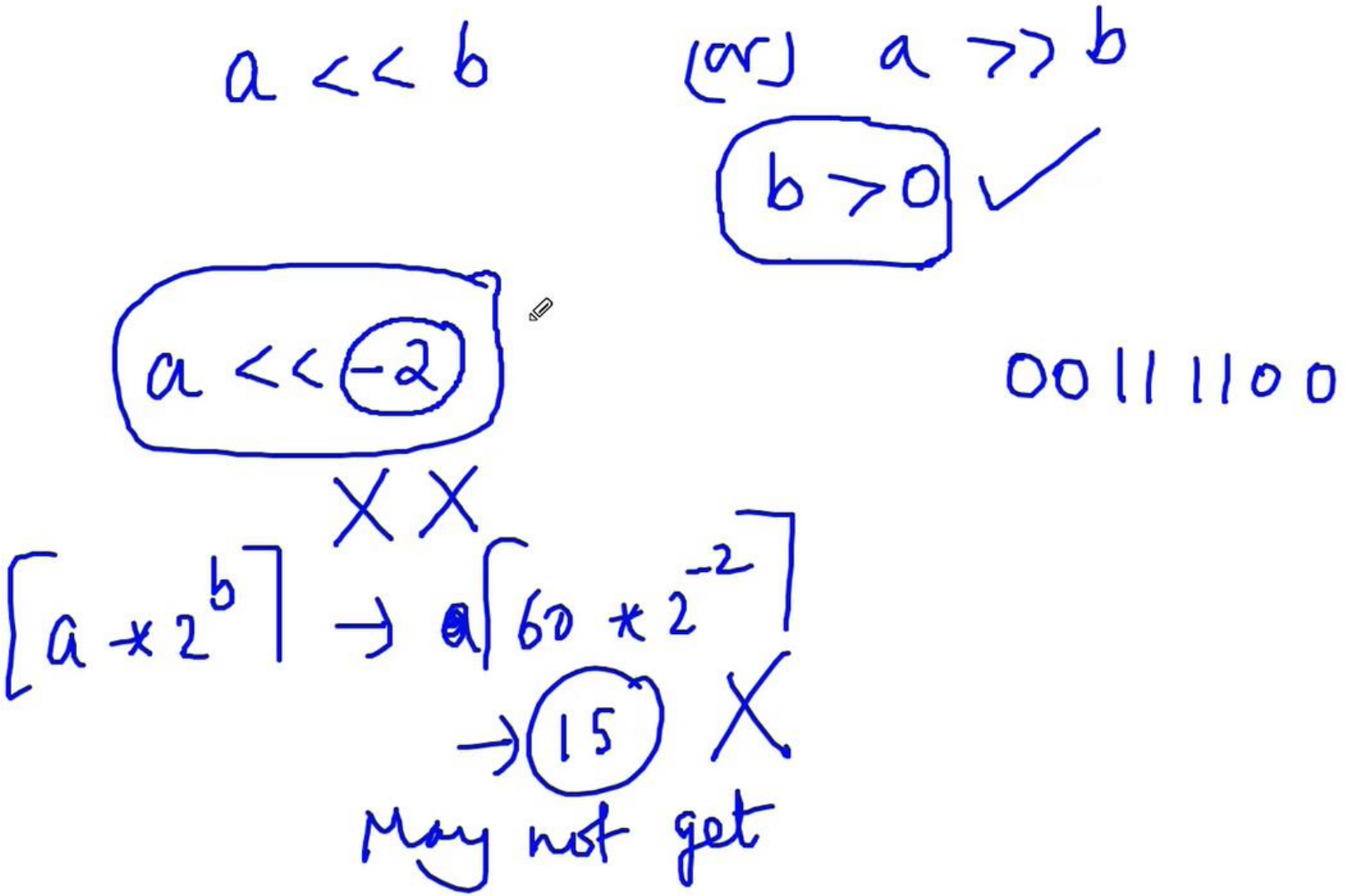
```
15
```

https://ideone.com/ieiOFK

Shift Operator



Shift Operator



SizeOfEx.c

```c
#include <stdio.h>
int main() {
int a = 600;
   printf("Size of variable a : %d\n",sizeof(a));
   printf("Size of int data type : %d\n",sizeof(int));
   printf("Size of char data type : %d\n",sizeof(char));
   printf("Size of float data type : %d\n",sizeof(float));
   printf("Size of double data type : %d",sizeof(double));
   return 0;
}
```

```
~$ ./a.out
Size of variable a : 4
Size of int data type : 4
Size of char data type : 1
Size of float data type : 4
Size of double data type : 8
~$
```

https://ideone.com/T6hsYB

The precedence and associativity of operators

Category	Operator	Associativity
Postfix	() [] -> . ++ - -	Left to right
Unary	+ - ! ~ ++ - - (type)* & sizeof	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Shift	<< >>	Left to right
Relational	< <= > >=	Left to right
Equality	== !=	Left to right
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right

The precedence and associativity of operators

Category	Operator	Associativity
Bitwise OR	|	Left to right
Logical AND	&&	Left to right
Logical OR	||	Left to right
Conditional	?:	Right to left
Assignment	= += -= *= /= %=>>= <<= &= ^= |=	Right to left
Comma	,	Left to right

Right to Left

</> source code

```c
#include <stdio.h>

int main(void) {
    // your code goes here
    int a=60;
    int b=2;
    int c=~a+b-~b;
    printf("%d",c);
    return 0;
}

```

stdout

```
-56
```

https://ideone.com/6CTCp0

Right to Left

</> source code

Success #stdin #stdout 0s 5432KB

4

```c
#include <stdio.h>

int main(void) {
    // your code goes here
    int a=3;
    int b=2;
    int c=1;
    int d=~a/b*~c;
    printf("%d",d);
    return 0;
}
```

https://ideone.com/g4sMPE

Right to Left

Right2Left.c

```c
#include <stdio.h>

int main(void) {
    // your code goes here
    int a=5;
    int b=6;
    a*=b*=5;
    printf("%d,%d\n", a,b);
    return 0;
}
```

```
~$ gcc Right2Left.c
~$ ./a.out
150,30
~$
```

https://ideone.com/COmvxh

C switch Statement

```
switch(expression){
case value1:
 //code to be executed;
 break;  //optional
case value2:
 //code to be executed;
 break;  //optional
......

default:
 code to be executed if all cases are not match
ed;
}
```

C switch Statement

</> source code

```c
#include <stdio.h>
int main()
{
    int x = 10, y = 5;
    switch(x>y && x+y>0)
    {
        case 1:
        printf("IITR");
        break;
        case 0:
        printf("IITM");
        break;
        default:
        printf("IITK");
    }

}

```

Success #stdin #stdout 0.01s 5516KB

```
IITR
```


https://ideone.com/MZKPsU

C switch Statement

```c
#include <stdio.h>

int main () {

   /* local variable definition */
   char grade = 'B';

   switch(grade) {
      case 'A' :
         printf("Excellent!\n" );
         break;
      case 'B' :
      case 'C' :
         printf("Well done\n" );
         break;
      case 'D' :
         printf("You passed\n" );
         break;
```

C switch Statement

```
19          case 'F' :
20              printf("Better try again\n" );
21              break;
22          default :
23              printf("Invalid grade\n" );
24      }
25
26      printf("Your grade is  %c\n", grade );
27
28      return 0;
29  }
30
```

stdout

```
Well done
Your grade is  B
```

https://ideone.com/eYrZ7L

C switch Statement

```c
#include <stdio.h>

int main () {

   /* local variable definition */
   char grade = 'B';

   switch(grade) {
      case 'A' :
         printf("Excellent!\n" );
         break;
      case 'B' :
      case 'C' :
         printf("Well done\n" );
      case 'D' :
         printf("You passed\n" );
      case 'F' :
         printf("Better try again\n" );
         break;
```

C switch Statement

```
20          default :
21              printf("Invalid grade\n" );
22      }
23
24      printf("Your grade is  %c\n", grade );
25
26      return 0;
27  }
28
```

https://ideone.com/3QEqen

stdout

```
Well done
You passed
Better try again
Your grade is  B
```

Rules

- The variable used in a switch statement can only be a short, int, or char.
- You can have any number of case statements within a switch. Each case is followed by the value to be compared to and a colon.
- The value for a case must be the same data type as the variable in the switch and it must be a constant or a literal.
- When the variable being switched on is equal to a case, the statements following that case will execute until a *break* statement is reached.

Rules

- When a *break* statement is reached, the switch terminates, and the flow of control jumps to the next line following the switch statement.

- Not every case needs to contain a break. If no break appears, the flow of control will *fall through* to subsequent cases until a break is reached.

- A *switch* statement can have an optional default case, which must appear at the end of the switch. The default case can be used for performing a task when none of the cases is true. No break is needed in the default case.

Decision Making and Looping

- The process of repeatedly executing a block of statements is known as looping
- The C language provides for three constructs for performing loop operations
  - `while` construct
  - `do` construct
  - `for` construct

The `while` Loop

- A while loop is a control structure that allows you to repeat a task a certain number of times.
- **Syntax**

```
while(Boolean_expression)
{
   //Statements
}
```

The while Loop

whileex1.c

```c
#include <stdio.h>

int main () {

   /* local variable definition */
   int a = 10;

   /* while loop execution */
   while( a < 20 ) {
      printf("value of a: %d\n", a);
      a++;
   }

   return 0;
}
```

```
~$ gcc whileex1.c
~$ ./a.out
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
~$
```

https://ideone.com/QosfjB

The while Loop

whileex2.c

```c
#include<stdio.h>
int main()
{
 int i=1,fact=1,number;
 printf("Enter a number: ");
  scanf("%d",&number);
    while (i<=number){
      fact=fact*i;
      i++;
  }
  printf("Factorial of %d is: %d\n",number,fact);
return 0;
}
```

```
~$ gcc whileex2.c
~$ ./a.out
Enter a number: 6
Factorial of 6 is: 720
~$
```

Thank You!