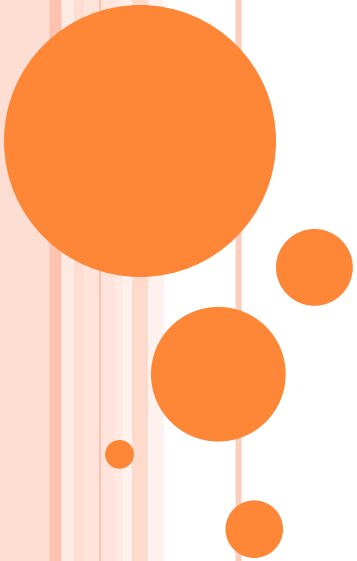


Loop And Bitwise Operators

Flow Control: Loop

- Sometime we need to repeat a set of action for several times
- For example you want to print ‘Happy birthday” for 10 times.
- We can do it using printf(“Happy birthday”) for 10 times.
- Is there exist any smarter solution?

- Loop is the smart solution for above problem
- We can control the number of repetition
- There are different constructs for looping in C
  - while, do..while, for

Flow Control – while

- `while (expr)`
  `stmt1`

- While the expression expr is TRUE execute statement stmt1. The while loop continues until expr becomes false. When expr becomes false the statement following stmt1 is executed.

Flowchart of while loop

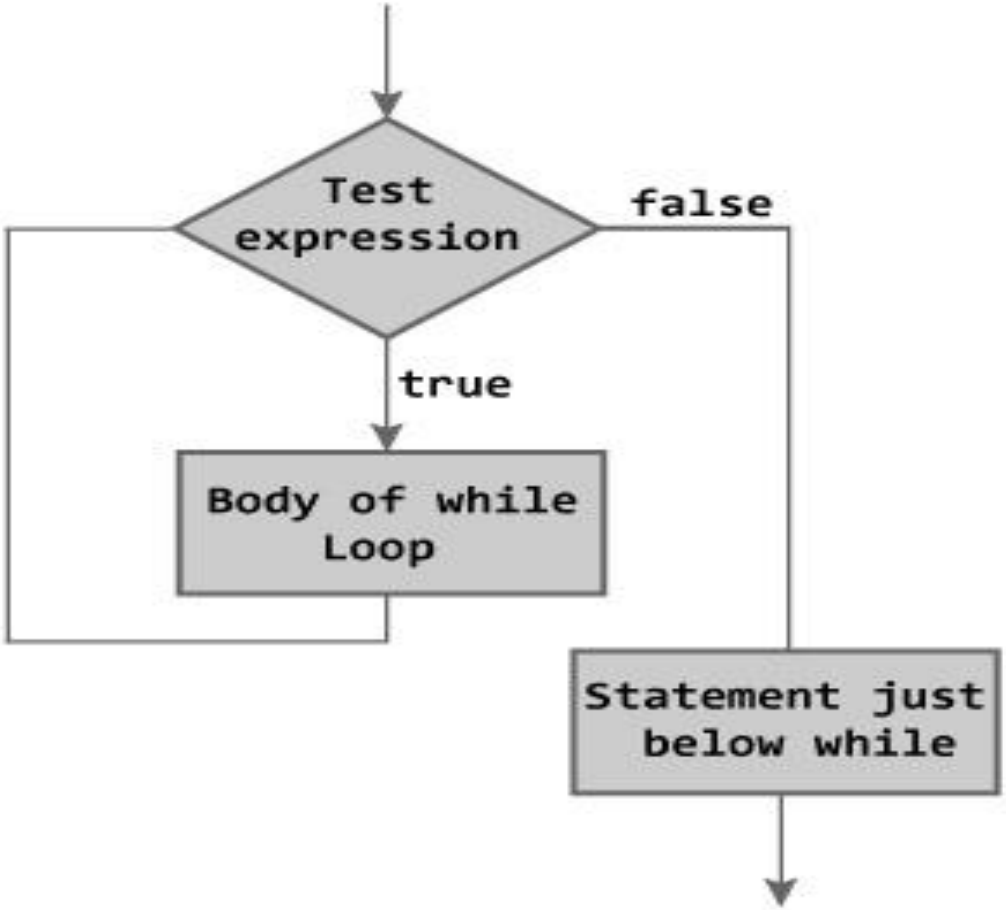


Figure: Flowchart of while Loop

Take an input N from user, write a program to print the values 1 to N.

```
#include<stdio.h>
int main(){
    int N,i=1;
    printf("Enter value of N\n");
    scanf("%d",&N);
    while(i<=N)
        printf("%d\n",i++)
    return 0;
}
```

- Program to find factorial of a number

```
#include <stdio.h>
int main() {
int number; long long factorial;
printf("Enter an integer: ");
scanf("%d",&number);
factorial = 1;
while (number > 0) {
    factorial *= number--;
}
printf("Factorial= %lld", factorial);
return 0; }
```

Assignment using while

- Write a C program that accepts n (read from keyboard) real numbers from the keyboard and prints out the difference of the maximum and minimum values of these numbers.

- Find the value of following series with accuracy up to 4 decimal places.
  - cos(x) = 1 - x^2/2! + x^4/4! -x^6/6! + ...

Flow Control – do .. while

- do

```
    stmt1
while (expr)
```

- While the expression expr is TRUE (nonzero) execute statement stmt1. The while loop continues until expr becomes false.

- What is the difference of the do .. while loop with that of the while loop?

Flowchart do-while loop

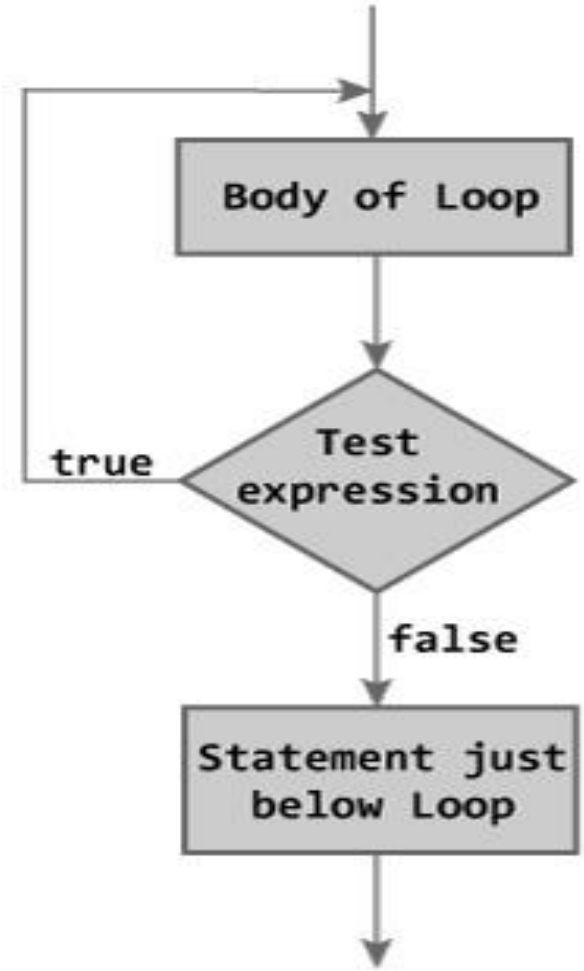


Figure: Flowchart of do...while Loop

- Program to add numbers until user enters zero
  #include <stdio.h>

```
int main() {
double number, sum = 0;
do {
    printf("Enter a number: ");
    scanf("%lf", &number);
    sum += number;
} while(number != 0.0);
printf("Sum = %lf",sum);
return 0; }
```

Assignment using do-while

- Write a C program that accepts n (read from keyboard) real numbers from the keyboard and prints out the difference of the maximum and minimum values of these numbers.

- Find the value of following series with accuracy up to 4 decimal places.
  - ln(1+x) = x - x^2/2 + x^3/3 -x^4/4 +

Flow Control – for

- Syntax: for (expr1; expr2; expr3)
  stmt1
  stmt2
- Any or all expression statements (exprs) can be missing.

FLOWCHART FOR LOOP

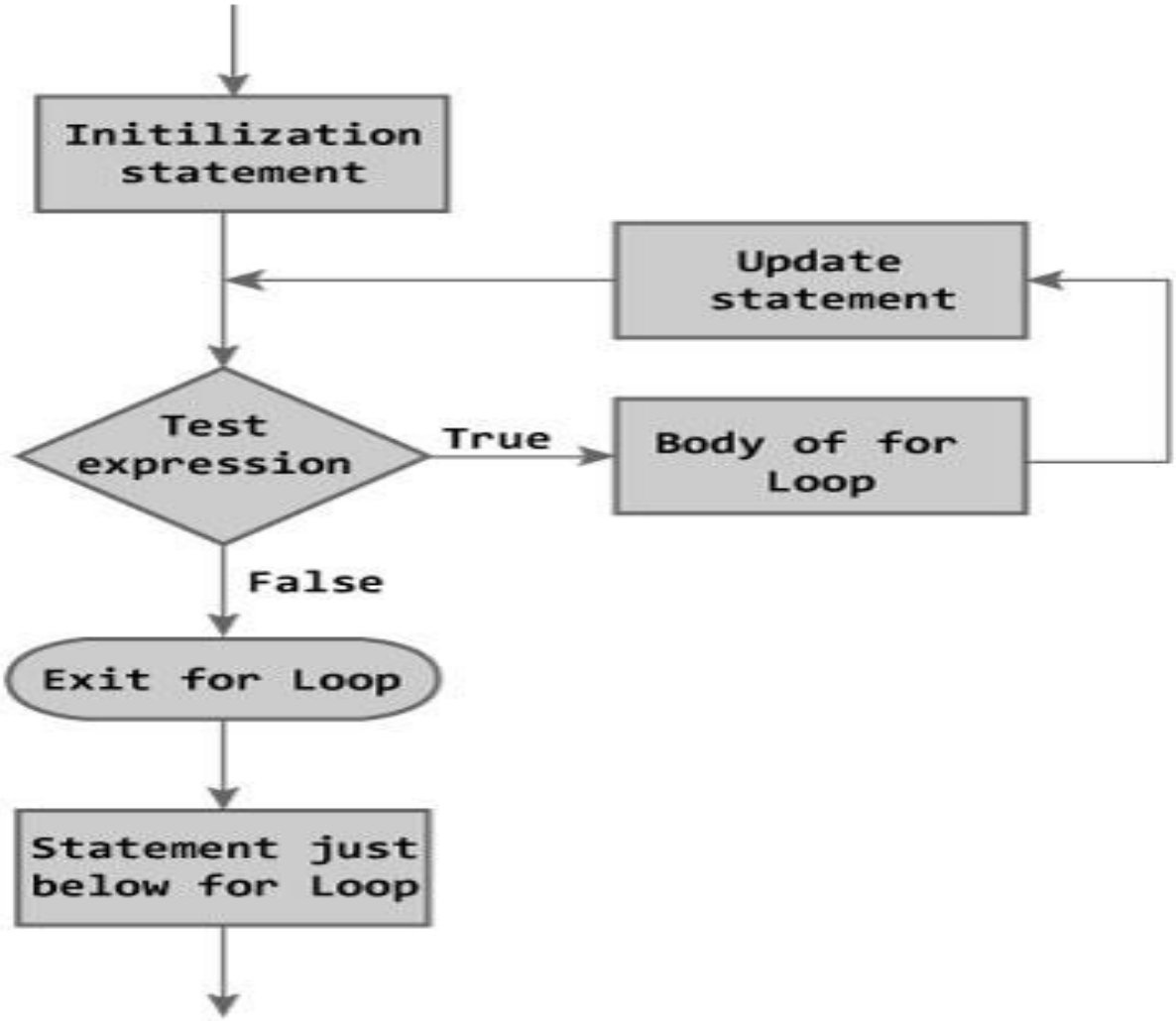


Figure: Flowchart of for Loop

- Program to print all odd numbers between 1 and N

```
for(i = 1; i <= N; i += 2)
        printf("%d", i);
```

- Program to calculate the sum of first n natural numbers

```
#include <stdio.h>
int main() {
int num, count, sum = 0;
printf("Enter a positive integer: ");
scanf("%d", &num);
for(count = 1; count <= num; ++count) {
    sum += count;
}
printf("Sum = %d", sum);
return 0; }
```

Assignment using do-while

- Read an input integer **x** from the keyboard and print the number of digits in x and the sum of all digits of that integer x. For example if the integer x is 456378 then your output should be x is a 6 digit number and sum of all digits in x is 33.

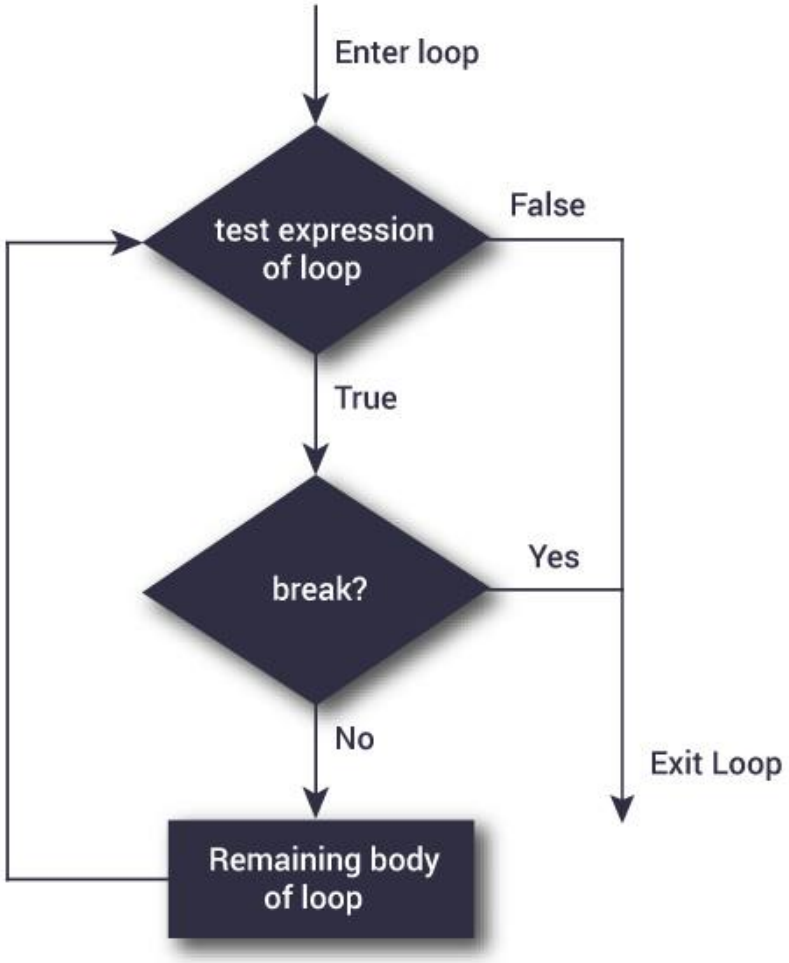
- Find the factorial of n

BREAK STATEMENT

- break statement causes to exit from the innermost enclosing loop or switch statement.
- Example:

```
while (1) {
        scanf("%f", &input);
        if (input < 0.0 )
            break;   }
```

Flowchart for break



Program to calculate the sum of maximum of 10 numbers ; Calculates sum until user enters positive number

```
# include <stdio.h>
int main() {
    int i; double number, sum = 0.0;
    for(i=1; i <= 10; ++i) {
        printf("Enter a n%d: ",i);
        scanf("%lf",&number);
        if(number < 0.0) { break; }
            sum += number;
    }
printf("Sum = %.2lf",sum);
return 0; }
```

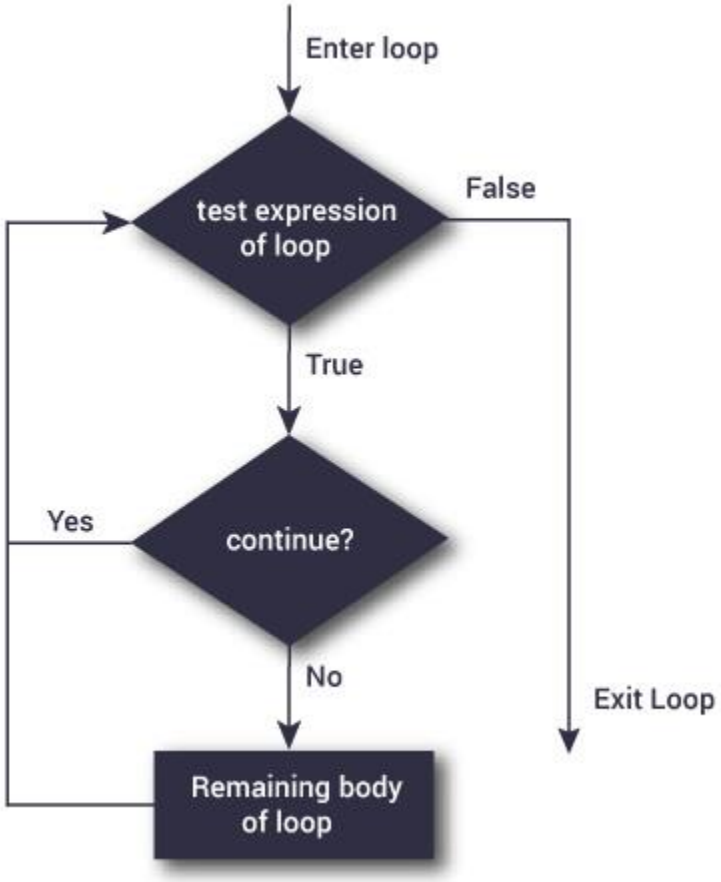
CONTINUE STATEMENT

```
while (1) {
    scanf("%f", &input);
      if (input < 0.0 ) {
          printf("Positive value only\n");
          continue;   }
     printf("%f\n",input);
 }
```

CONTINUE STATEMENT

- May occur only inside for, while and do loops.
- Causes to skip the remaining statement of the loop and continues with the next iteration of the loop.

Flowchart for continue



Program to calculate sum of maximum of 10 numbers ;Negative numbers are skipped from calculation

```
# include <stdio.h>
int main() {
    int i; double number, sum = 0.0;
    for(i=1; i <= 10; ++i) {
        printf("Enter a n%d: ",i);
        scanf("%lf",&number);
        if(number < 0.0) {
            continue;
        }
        sum += number;
    }
    printf("Sum = %.2lf",sum);
    return 0;
}
```

Flow Control - goto

- Causes unconditional jump to a labeled statement.

- Syntax: label: Statement

```
begin: for (i = 1; i <= 10; i++) {
        if ( i == 5 )
                goto begin;
        printf("%d", i);                }
```

What is Bitwise Structure?

- The smallest type is of 8 bits (char).
- Sometimes we need only a single bit.
- For instance, storing the status of the pass/fail in 8 subjects:
  - We need to define an array of at least 8 chars.
    If a student passed in 3rd subject then corresponding array position has to be set
  - Total memory requires for storing is 64 bits.

What is Bitwise Structure?

- It is better to define only 8 bits since a bit can also store the values 0 or 1.
- But the problem is that there is no C type which is 1 bit long (char is the longer with 1 byte).
- Solution: define a char (8 bits) but refer to each bit separately.
- **Bitwise** operators, introduced by the C language, provide one of its more powerful tools for using and manipulating memory. They give the language the real power of a "low-level language".

What is Bitwise Structure?

- A single bit cannot be accessed directly, since it has no address of its own.
- The language introduces the **`bitwise`** operators, which help in manipulating a single bit of a byte.
- **`bitwise`** operators may be used on integral types only (unsigned types are preferable).

Bitwise Operators

&	bitwise AND
|	bitwise OR
^	bitwise XOR
~	1's compliment
<<	Shift left
>>	Shift right

Bitwise Operators – truth table

a	b	a&b	a|b	a^b	~a
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

Bitwise Operators - Examples

```
11010011
&
10001100
-----------
10000000
```

```
11010011
|
10001100
-----------
11011111
```

```
11010011
^
10001100
-----------
01011111
```

```
~11010011
-----------
  00101100
```

```
11010011>>3
-----------
  00011010
```

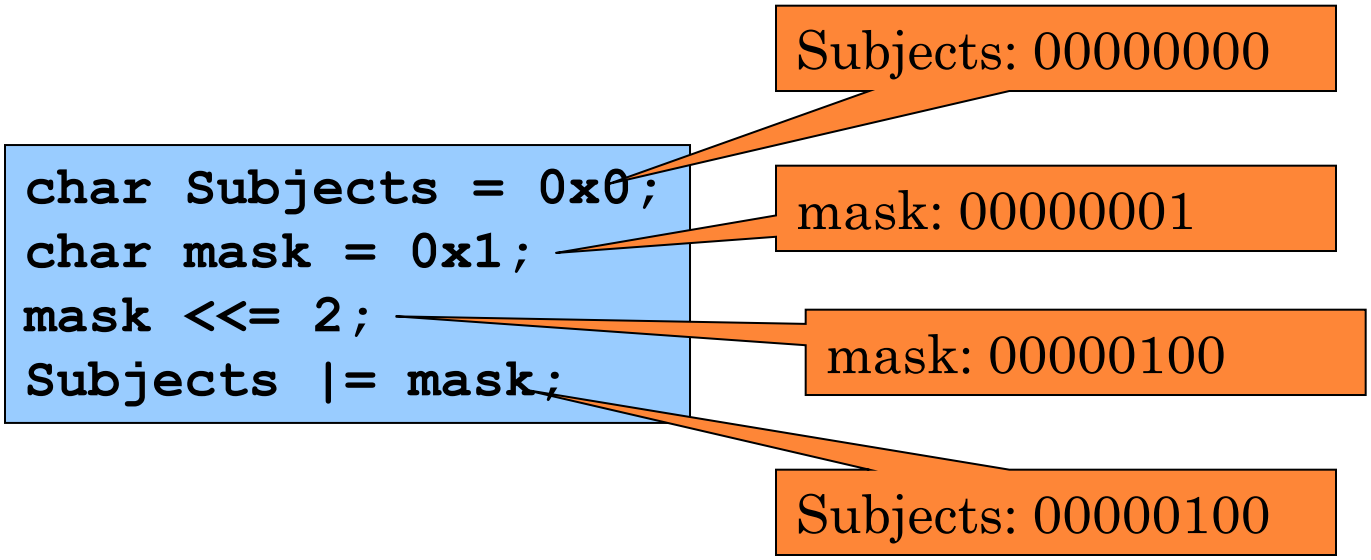
```
11010011<<3
-----------
  10011000
```

Setting Bits

- How can we set a bit on or off?
- Manipulations on bits are enabled by mask and bitwise operators.
- Bitwise OR of anything with 1 results in 1.
- Bitwise AND of anything with 0 results in 0.

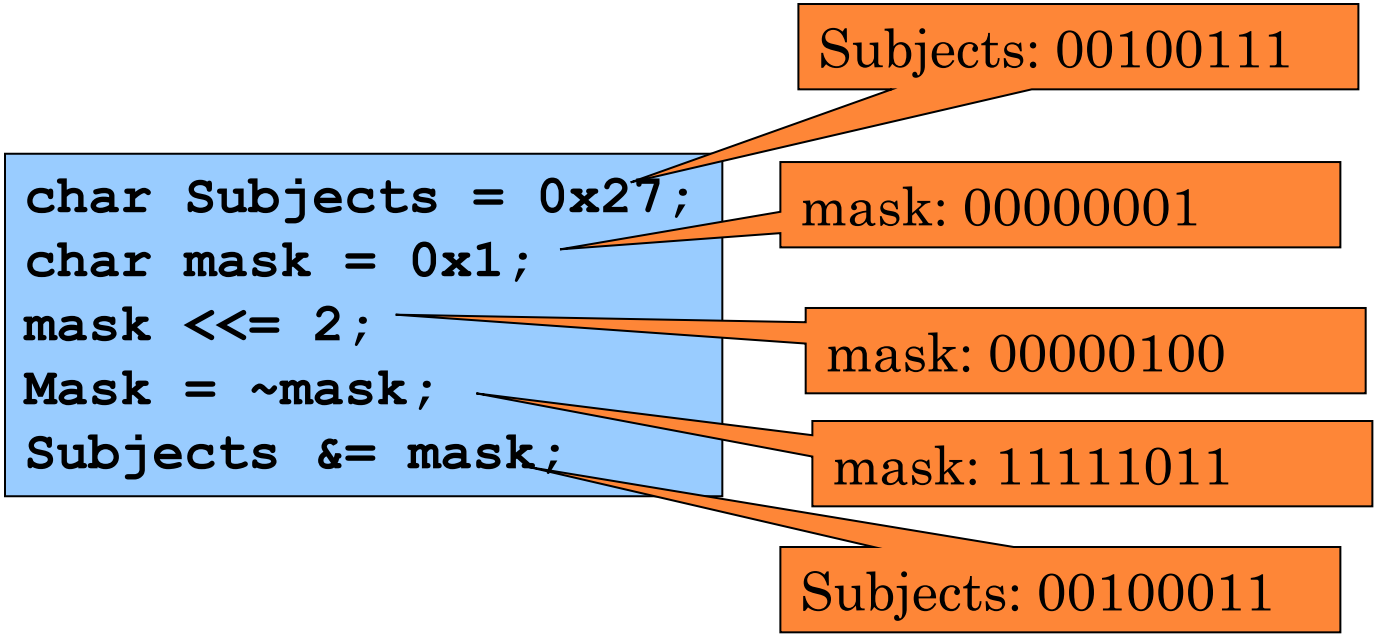
Setting Bits

- For instance, how can we set the bit no #3?



Turn off Bits

- For instance, how can we turn off the bit no #3?

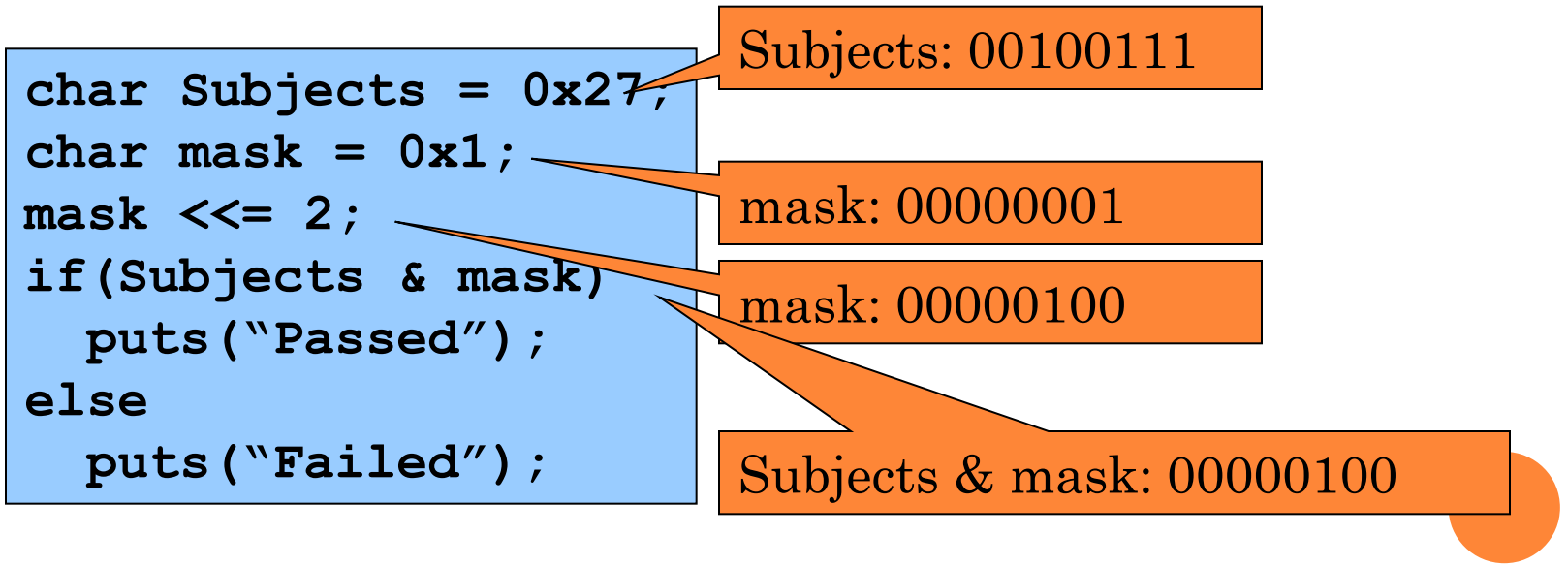


Getting Bits

- How can we know if a bit is on or off?
- Manipulations on bits are enabled by mask and bitwise operators.
- Bitwise AND of anything with 1 results in the same value.

Getting Bits

- For instance, how can we check if a student passed in subject #3?



Bitwise - Example

Suppose we have 8 Subjects:

- A student passed in certain subjects.
- We like to know which subjects student passed.

```
void main()
{
   unsigned char Subjects = 0;
   set_Subjects
   print_status
}
```

```
#include<stdio.h>

int main(void){
unsigned char Subjects=0;
int j, answer;
unsigned char mask;
for(j=0,mask=1; j<8; j++,mask<<=1)
{
        answer=0;
        printf(“Enter non-zero if you passed or zero if you failed in
subject #%d\n", j+1);
        scanf("%d“,&answer);
        if(answer)
                Subjects |= mask;
}
printf(“Entered status of pass-fail is %d\n",Subjects);
```

```
for(j=0,mask=1; j<8; j++,mask<<=1)
 {
  if(Subjects & mask)
   printf("You passed in #%d Subject\n",j+1);
  else
   printf ("You failed in #%d Subject\n",j+1);
 }


return 0;
}
```

Assignment

- Let's say students have 8 courses in a semester and subjects are 1st, 2nd,...,8th. Student's pass/fail status on all subjects can be understood from a code say a student who passed in all subjects except 5th will get the binary code 11101111 or it's corresponding integer representation 239. So score can vary in the range of 0 to 255. Read scores of two students say Ram and Varun and compute
  - Number of subjects Ram passed
  - Number of subjects where at least one of them passed
  - Number of subjects in which only Ram Passed but Varun Failed
  - Number of subjects in which both passed
  - Number of subjects in which their passing status differ

- #include<stdio.h>

```
int main(void){
int ram_score;
unsigned char ram_score_bin=0, mask;
int j,number_of_pass = 0;
printf("Enter marks of Ram \n");
scanf("%d",&ram_score);
for(j=0,mask=1;j<8;j++,mask<<=1){
        if(ram_score%2)
                ram_score_bin |= mask;
        ram_score=ram_score/2;
}
```

```
for(j=0,mask=1; j<8; j++,mask<<=1)
 {
  if(ram_score_bin & mask)
        number_of_pass ++;
 }
printf("Number of subjects Ram passed is
%d\n",number_of_pass);

return 0;
}
```

```
#include<stdio.h>

int main(void){
int ram_score,varun_score;
unsigned char ram_score_bin=0,
varun_score_bin,at_least,mask;
int j,number_of_pass = 0;
printf("Enter marks of Ram and Varun\n");
scanf("%d%d",&ram_score,&varun_score);
for(j=0,mask=1;j<8;j++,mask<<=1){
        if(ram_score%2)
                ram_score_bin |= mask;
        if(varun_score%2)
                varun_score_bin |= mask;
        ram_score=ram_score/2;
        varun_score=varun_score/2;
}
```

```
at_least = ram_score_bin | varun_score_bin;
for(j=0,mask=1; j<8; j++,mask<<=1)
 {
  if(at_least & mask)
        number_of_pass ++;
 }
printf("Number of subjects at least one passed is %d\n",number_of_pass);


return 0;
}
```

```
#include<stdio.h>

int main(void){
int ram_score,varun_score;
unsigned char ram_score_bin=0,
varun_score_bin,only_ram,mask;
int j,number_of_pass = 0;
printf("Enter marks of Ram and Varun\n");
scanf("%d%d",&ram_score,&varun_score);
printf("Marks of ram and varun are %d and
%d\n",ram_score,varun_score);
for(j=0,mask=1;j<8;j++,mask<<=1){
        if(ram_score%2)
                ram_score_bin |= mask;
        if(varun_score%2)
                varun_score_bin |= mask;
        ram_score=ram_score/2;
        varun_score=varun_score/2;
}
```

```
for(j=0,mask=1; j<8; j++,mask<<=1)
 {
  if((ram_score_bin & mask)&& (!(varun_score_bin
& mask)))
        number_of_pass ++;
 }
printf("Number of subjects where Ram  passed but
not Varun is %d\n",number_of_pass);



return 0;
}
```