Selection and Iterative Statements in C

Browsing Problem

Given the number of hours and minutes browsed, write a program to calculate bill for Internet Browsing in a browsing center. The conditions are given below.

(a) 1 Hour Rs.50

(b) 1 minute Re. 1

(c) Rs. 200 for five hours

Boundary condition: User can only browse for a maximum of 7 hours

Check boundary conditions

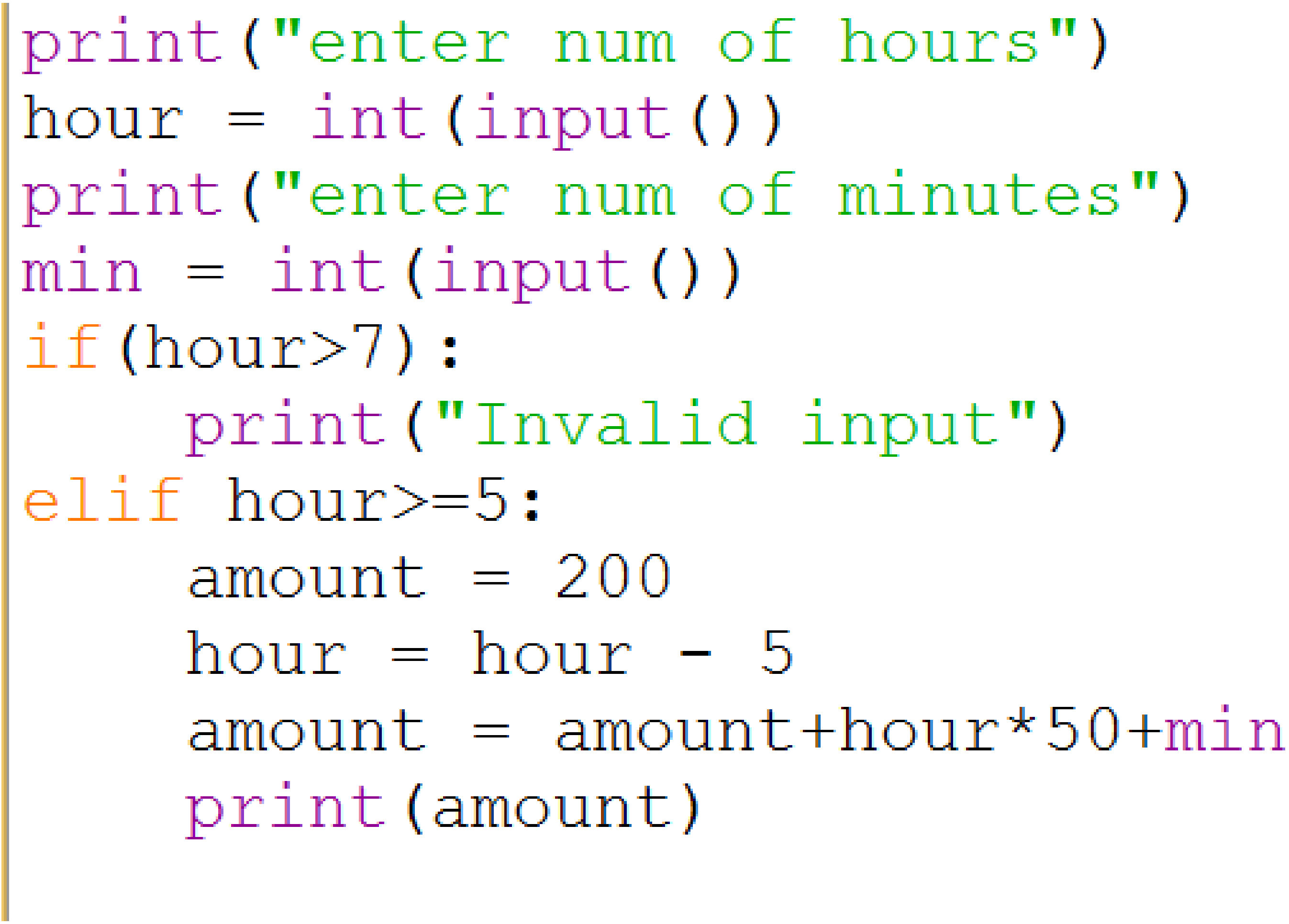
Browsing Problem

Input	Processing	Output
Number of hours and minutes browsed	Check number of hours browsed, if it is greater than 5 then add Rs 200 to amount for five hours and subtract 5 from hours Add Rs for each hour and Re 1 for each minute Basic process involved: Multiplication and addition	Amount to be paid

Pseudocode

```
READ hours and minutes
SET amount = 0
IF hours >=5 then
    CALCULATE amount as amount + 200
    COMPUTE hours as hours – 5
END IF
COMPUTE amount as amount + hours * 50
COMPUTE amount  as amount + minutes * 1
PRINT amount
```

Browsing Program



```
print("enter num of hours")
hour = int(input())
print("enter num of minutes")
min = int(input())
if(hour>7):
    print("Invalid input")
elif hour>=5:
    amount = 200
    hour = hour - 5
    amount = amount+hour*50+min
    print(amount)
```

Already you Know

- To read values from user
- Write arithmetic expressions in C
- Print values in a formatted way

Yet to Learn

- Check a condition

Syntax

Form 1:

if (condition)

statement $_T$;

Eg:

if (x > 0.0)

pos_prod = pos_prod * x;

If condition evaluates to true (a nonzero value), then

statement $_T$ is executed; otherwise, statement $_T$ is skipped.

Syntax

Form 2:

```
 if ( condition )
statement T  ;
else
statement F  ;
```

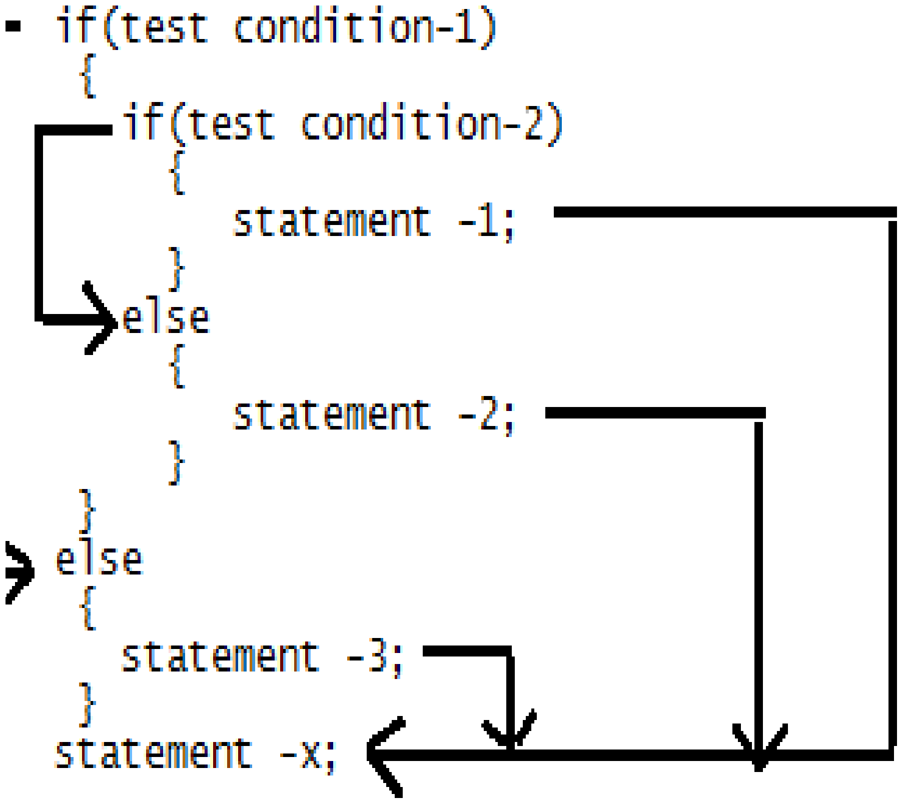
If condition evaluates to true (a nonzero value), then statement $_T$ is executed; otherwise, statement $_F$ is executed

Example

```
 if (x >= 0.0)

printf("positive\n");

else

printf("negative\n");
```

If statement in Python & C

```
if test condition-1:
    statement(s)
elif test condition-2:
    statement(s)
else:
    statement(s)
```



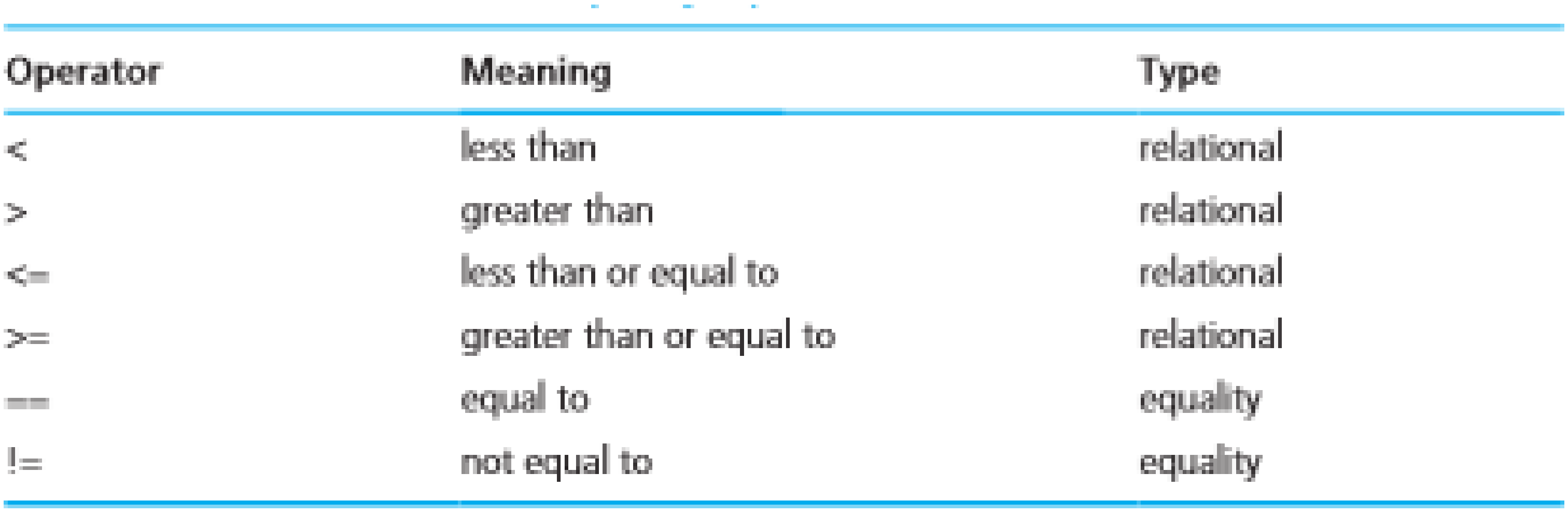
```
if(test condition-1)
 {
   if(test condition-2)
     {
        statement -1;
     }
   else
     {
        statement -2;
     }
 }
else
 {
   statement -3;
 }
statement -x;
```

Compound Statements

- Until now we have been using only sequential flow
- A compound statement, written as a group of statements bracketed by { and } , is used to specify sequential flow.

-
  ```
  {
   statement  1 ;
   statement  2 ;
   ...
   statement  n ;
  }
  ```

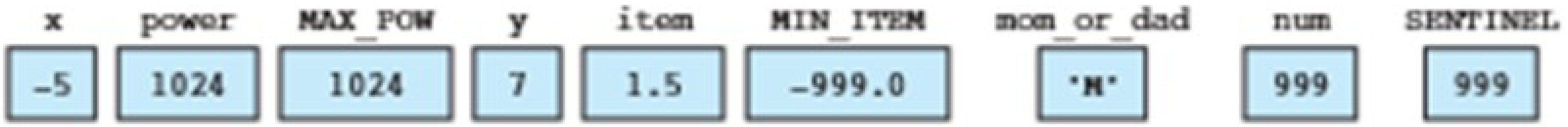
Relational and Equality Operators



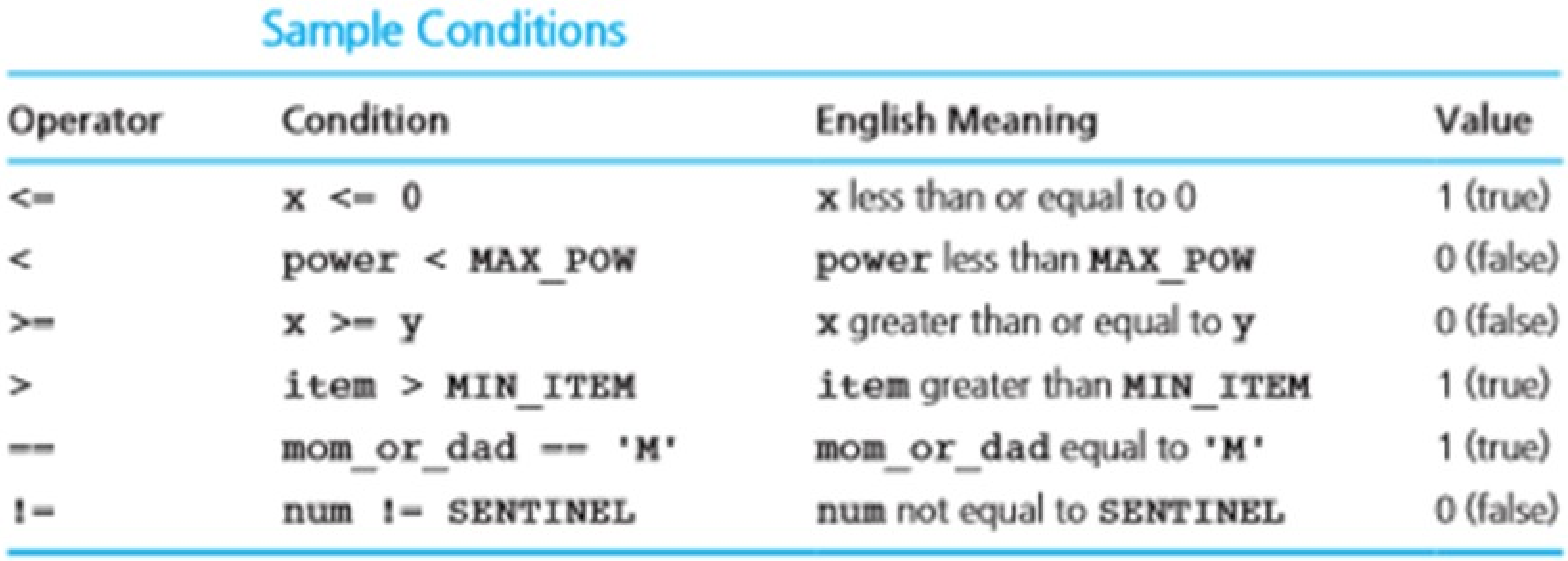
Operator	Meaning	Type
<	less than	relational
>	greater than	relational
<=	less than or equal to	relational
>=	greater than or equal to	relational
==	equal to	equality
!=	not equal to	equality

Examples

Memory with Values



x	power	MAX_POW	y	item	MIN_ITEM	mom_or_dad	num	SENTINEL
-5	1024	1024	7	1.5	-999.0	'M'	999	999



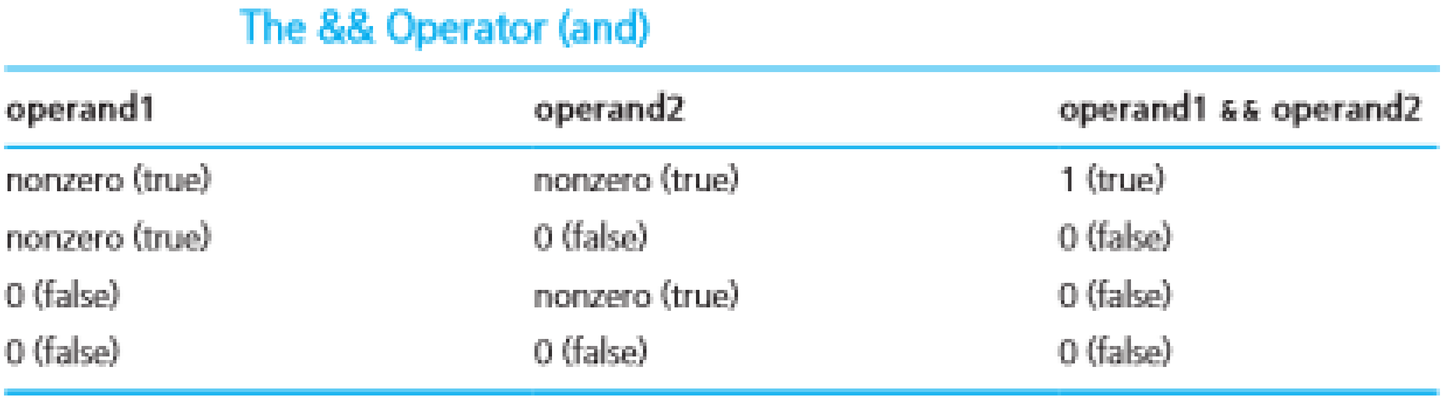
Sample Conditions

Operator	Condition	English Meaning	Value
<=	x <= 0	x less than or equal to 0	1 (true)
<	power < MAX_POW	power less than MAX_POW	0 (false)
>=	x >= y	x greater than or equal to y	0 (false)
>	item > MIN_ITEM	item greater than MIN_ITEM	1 (true)
==	mom_or_dad == 'M'	mom_or_dad equal to 'M'	1 (true)
!=	num != SENTINEL	num not equal to SENTINEL	0 (false)

Logical Operators

- To form more complicated conditions or logical expressions
- Three operators:
  - And (&&)
  - Or (||)
  - Not(!)

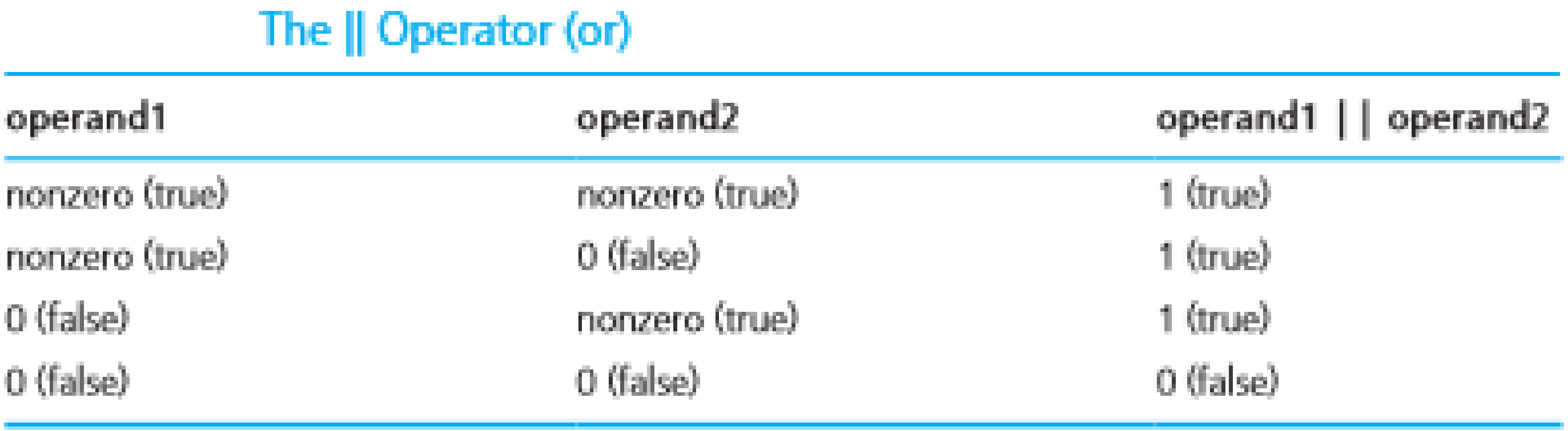
Logical and



The && Operator (and)

operand1	operand2	operand1 && operand2
nonzero (true)	nonzero (true)	1 (true)
nonzero (true)	0 (false)	0 (false)
0 (false)	nonzero (true)	0 (false)
0 (false)	0 (false)	0 (false)

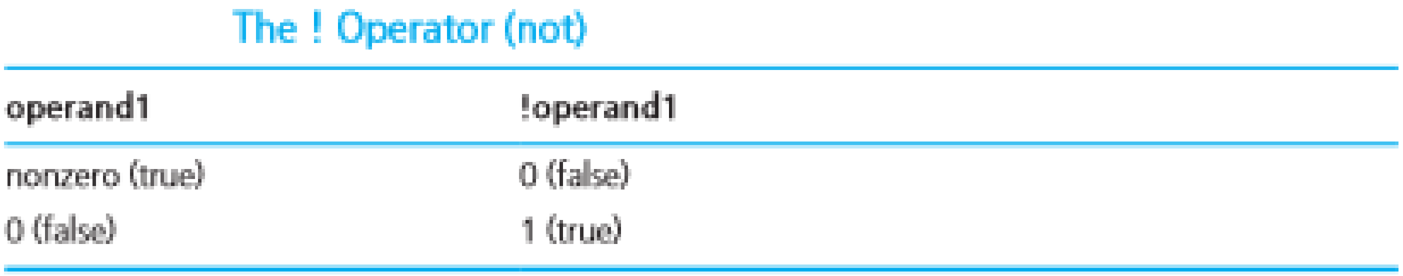
Logical Or



The || Operator (or)

operand1	operand2	operand1 | | operand2
nonzero (true)	nonzero (true)	1 (true)
nonzero (true)	0 (false)	1 (true)
0 (false)	nonzero (true)	1 (true)
0 (false)	0 (false)	0 (false)

Logical Not



The ! Operator (not)

operand1	!operand1
nonzero (true)	0 (false)
0 (false)	1 (true)

True/False Values

- For numbers all values except 0 is true
- For characters all values except ‘/0’ (Null Character) is true

Short Circuit Evaluation

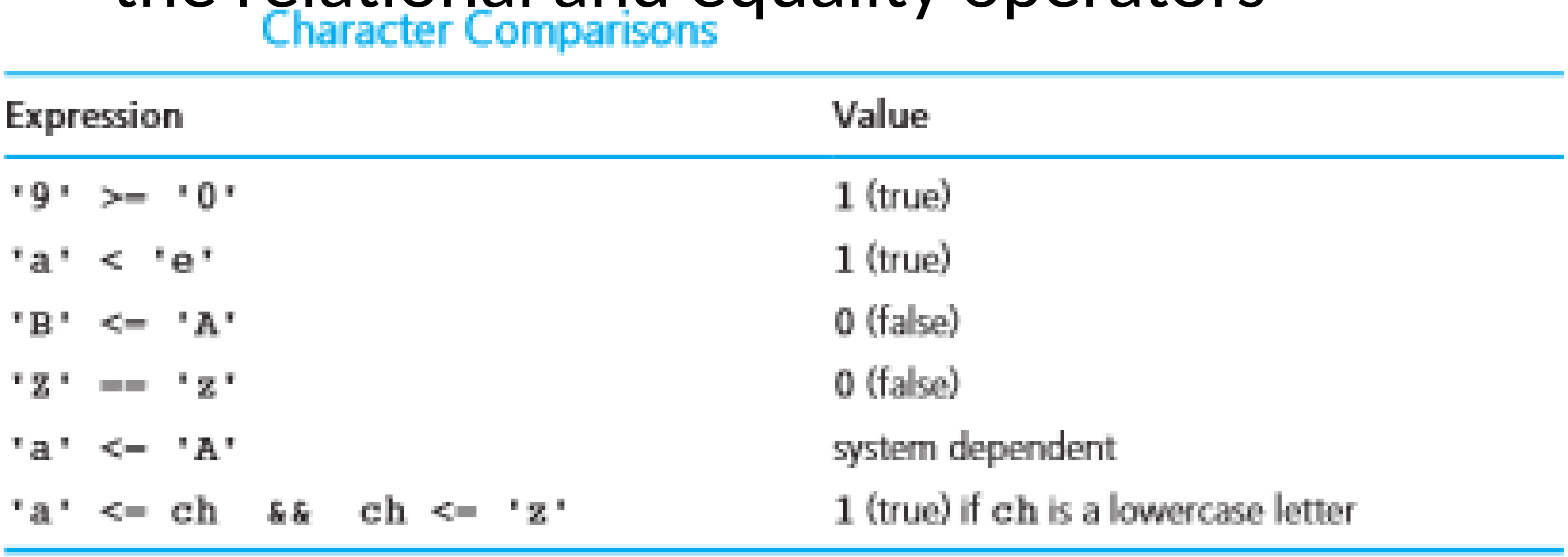
- Stopping evaluation of a logical expression as soon as its value can be determined is called short-circuit evaluation
- Second part of ‘&&’ does not gets evaluated when first part is evaluated as False
- Second part of ‘||’ does not gets evaluated when first part is evaluated as true

Short Circuit Evaluation

- (num % div == 0) – Runtime error if div = 0
- But prevented when written as
- (div != 0 && (num % div == 0))

Comparing Characters

- We can also compare characters in C using the relational and equality operators



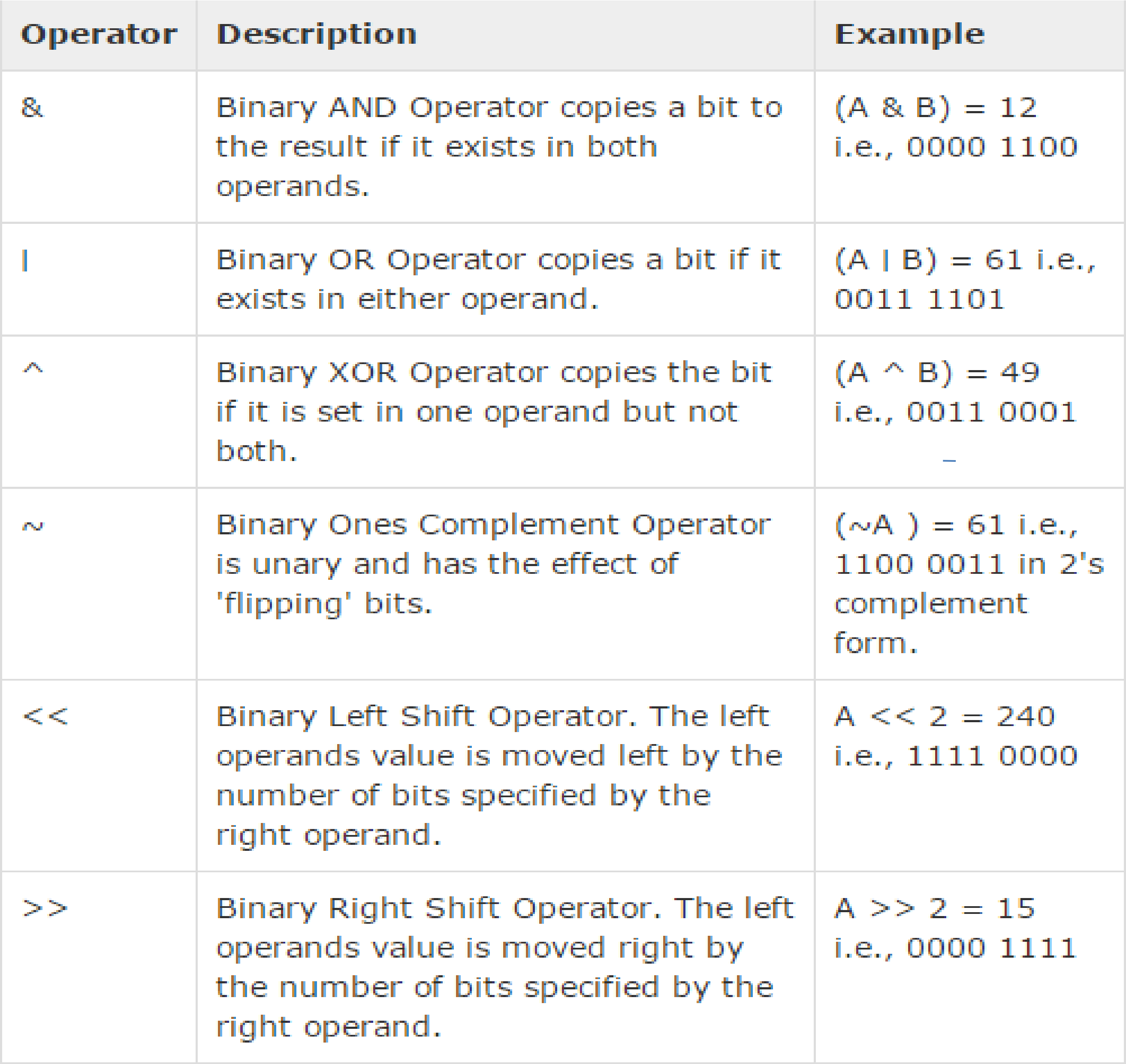
Character Comparisons

Expression	Value
`'9' >= '0'`	1 (true)
`'a' < 'e'`	1 (true)
`'B' <= 'A'`	0 (false)
`'Z' == 'z'`	0 (false)
`'a' <= 'A'`	system dependent
`'a' <= ch && ch <= 'z'`	1 (true) if ch is a lowercase letter

Logical Assignment

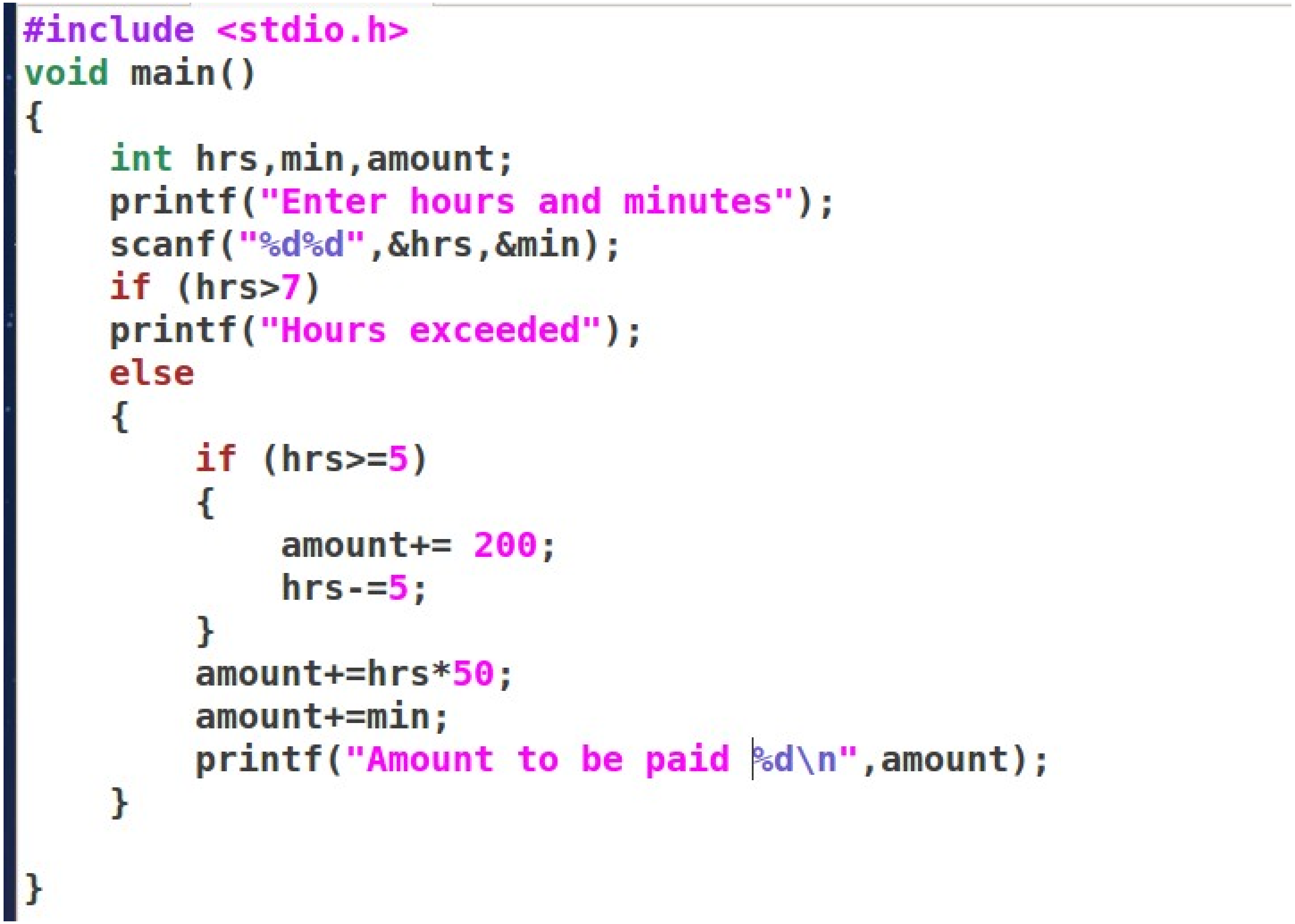
- even = (n % 2 == 0);
- in_range = (n > -10 && n < 10);
- is_letter = ('A' <= ch && ch <= 'Z') || ('a' <= ch && ch <= 'z');
- Variable in_range gets 1 (true) if the value of n is between −10 and 10 excluding the endpoints;
- is_letter gets 1 (true) if ch is an uppercase or a lowercase letter.

When 'A‘ = 60 and 'B' =13

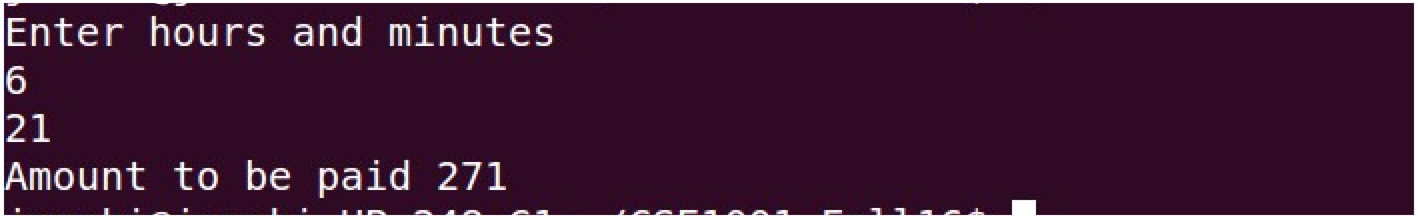


Operator	Description	Example
&	Binary AND Operator copies a bit to the result if it exists in both operands.	(A & B) = 12 i.e., 0000 1100
|	Binary OR Operator copies a bit if it exists in either operand.	(A | B) = 61 i.e., 0011 1101
^	Binary XOR Operator copies the bit if it is set in one operand but not both.	(A ^ B) = 49 i.e., 0011 0001
~	Binary Ones Complement Operator is unary and has the effect of 'flipping' bits.	(~A) = 61 i.e., 1100 0011 in 2's complement form.
<<	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.	A << 2 = 240 i.e., 1111 0000
>>	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.	A >> 2 = 15 i.e., 0000 1111

Example



```
#include <stdio.h>
void main()
{
    int hrs,min,amount;
    printf("Enter hours and minutes");
    scanf("%d%d",&hrs,&min);
    if (hrs>7)
    printf("Hours exceeded");
    else
    {
        if (hrs>=5)
        {
            amount+= 200;
            hrs-=5;
        }
        amount+=hrs*50;
        amount+=min;
        printf("Amount to be paid %d\n",amount);
    }

}
```

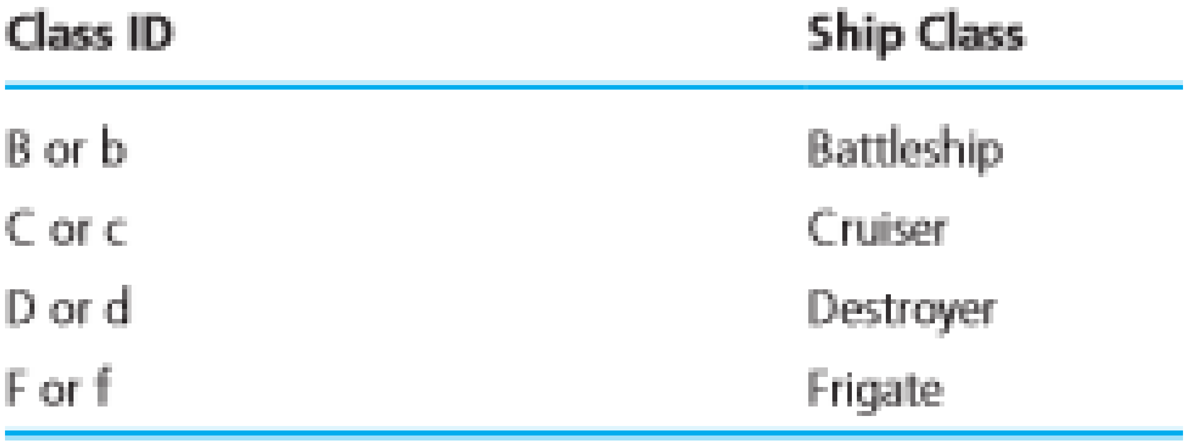


```
Enter hours and minutes
6
21
Amount to be paid 271
```

Example

```
/* 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;
```

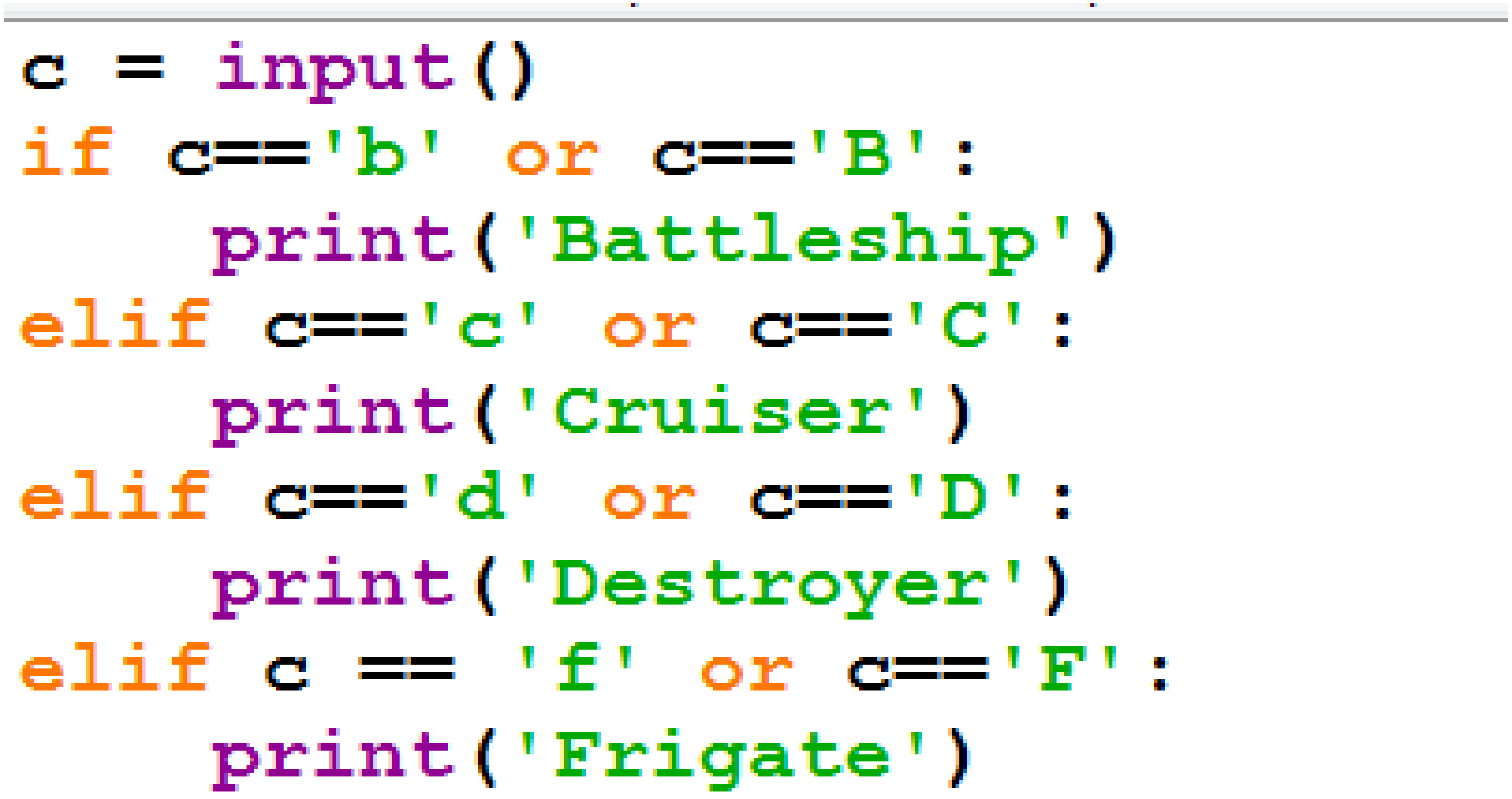
Class of the Ship



Class ID	Ship Class
B or b	Battleship
C or c	Cruiser
D or d	Destroyer
F or f	Frigate

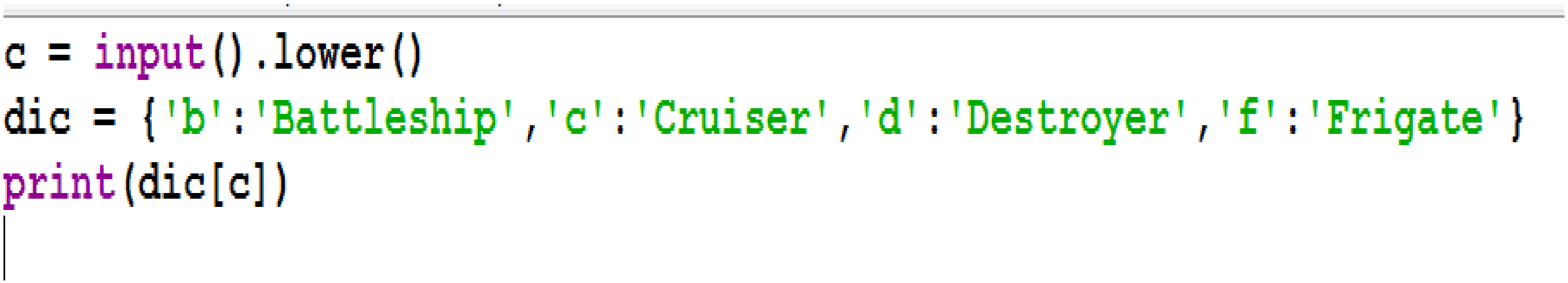
- Each ship serial number begins with a letter indicating the class of the ship. Write a program that reads a ship's first character of serial number and displays the class of the ship.

Program in C



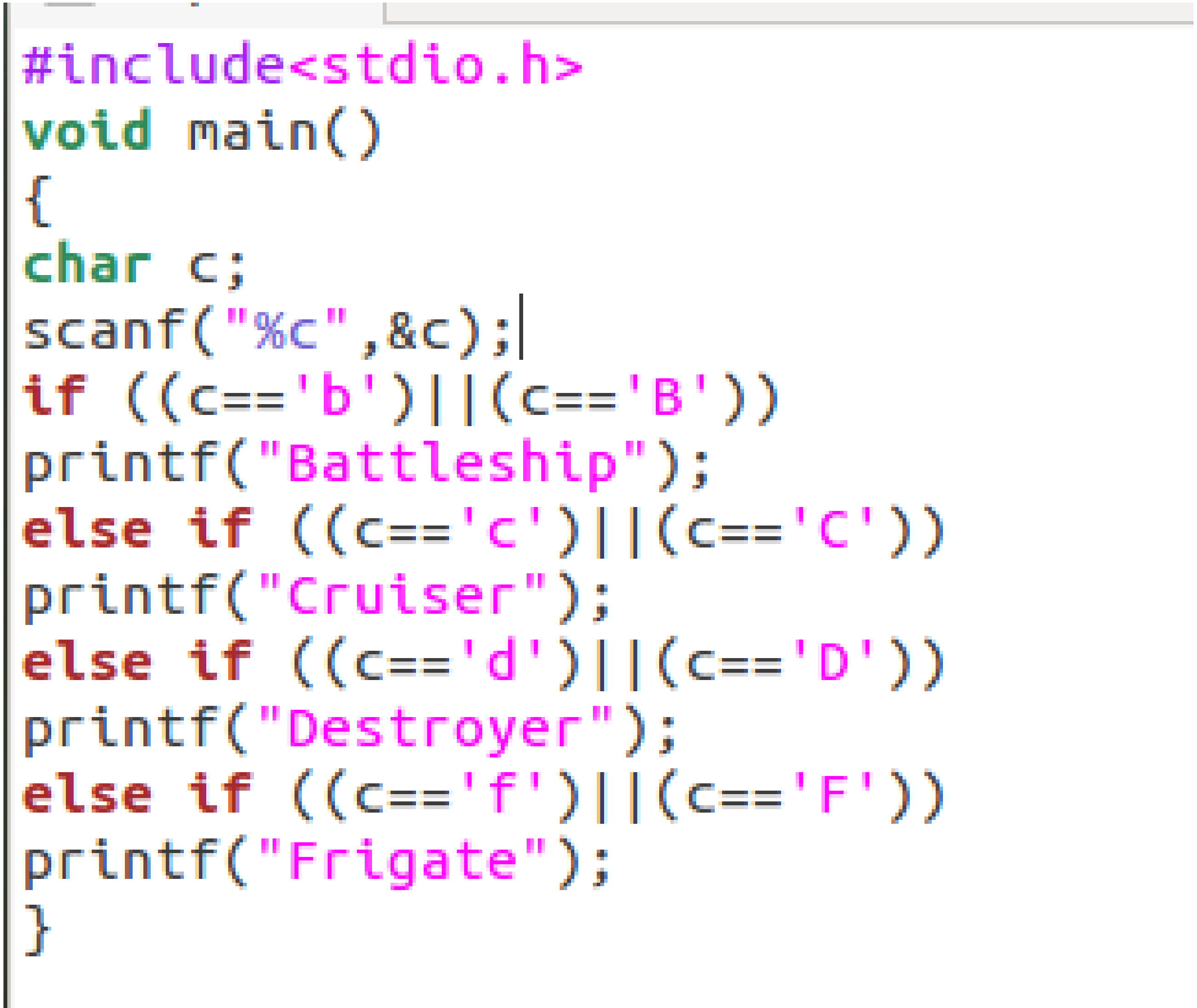
```
c = input()
if c=='b' or c=='B':
    print('Battleship')
elif c=='c' or c=='C':
    print('Cruiser')
elif c=='d' or c=='D':
    print('Destroyer')
elif c == 'f' or c=='F':
    print('Frigate')
```

Program in Python



```
c = input().lower()
dic = {'b':'Battleship','c':'Cruiser','d':'Destroyer','f':'Frigate'}
print(dic[c])
```

Nested If Statement



```
#include<stdio.h>
void main()
{
char c;
scanf("%c",&c);
if ((c=='b')||(c=='B'))
printf("Battleship");
else if ((c=='c')||(c=='C'))
printf("Cruiser");
else if ((c=='d')||(c=='D'))
printf("Destroyer");
else if ((c=='f')||(c=='F'))
printf("Frigate");
}
```

Switch Statement

- Useful when the selection is based on the value of a single variable or of a simple expression (called the controlling expression)
- Value of this expression may be of type int or char , but not of type double

Syntax of Switch

switch (controlling expression)

{

label set$_1$

statements$_1$

break;

label set$_n$

statements$_n$

break;

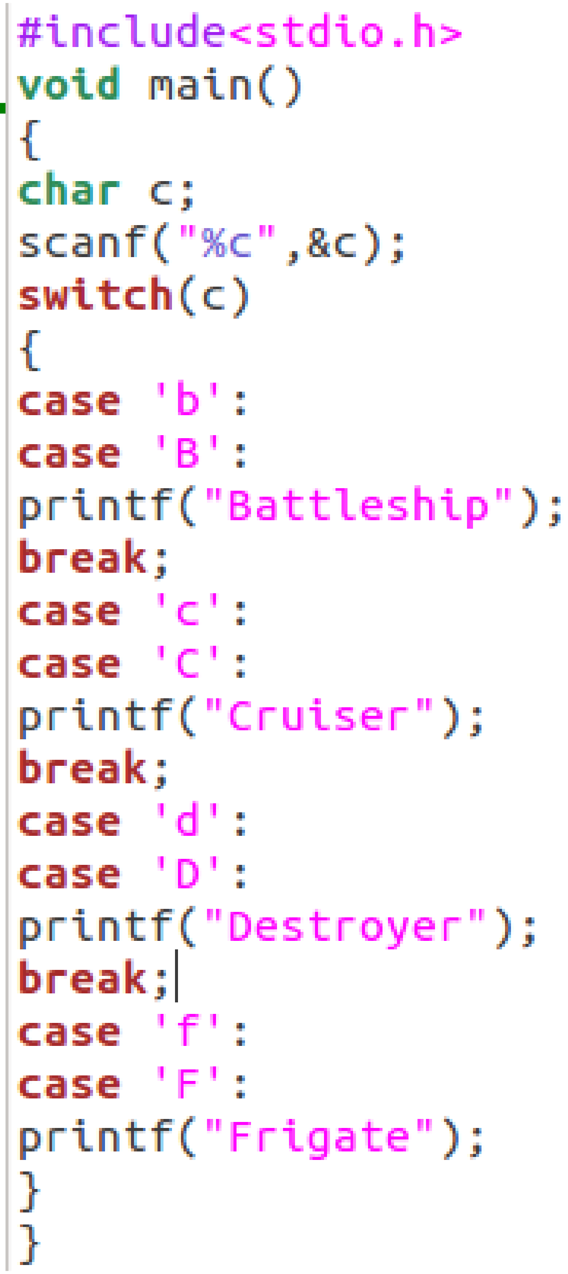
default:

statements$_d$

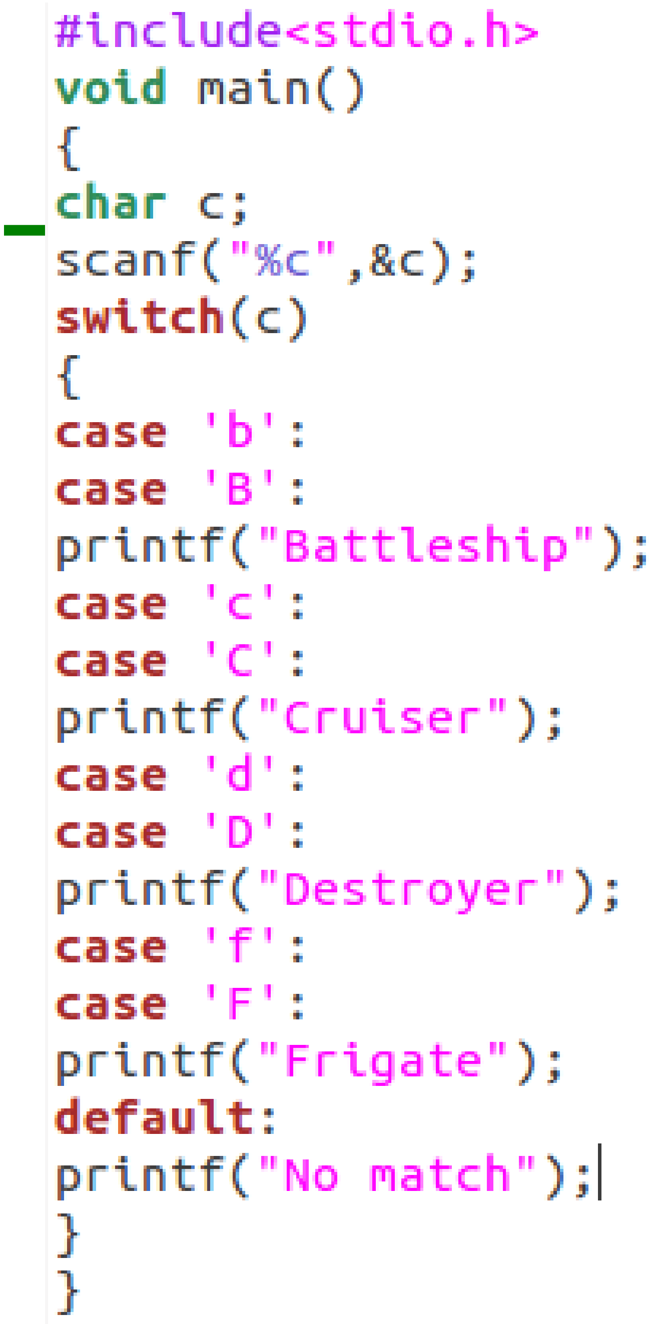
}

Syntax of Switch

- When a match between the value of the controlling expression and a case label value is found, the statements following the case label are executed until a break statement is encountered.
- Then the rest of the switch statement is skipped.
- If no case label value matches the controlling expression, the entire switch statement body is skipped unless it contains a default label.



```
#include<stdio.h>
void main()
{
char c;
scanf("%c",&c);
switch(c)
{
case 'b':
case 'B':
printf("Battleship");
break;
case 'c':
case 'C':
printf("Cruiser");
break;
case 'd':
case 'D':
printf("Destroyer");
break;
case 'f':
case 'F':
printf("Frigate");
}
}
```



```
#include<stdio.h>
void main()
{
char c;
scanf("%c",&c);
switch(c)
{
case 'b':
case 'B':
printf("Battleship");
case 'c':
case 'C':
printf("Cruiser");
case 'd':
case 'D':
printf("Destroyer");
case 'f':
case 'F':
printf("Frigate");
default:
printf("No match");
}
}
```

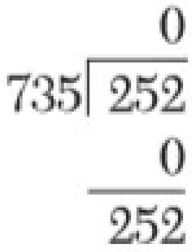
Output

BattleshipCruiserDestroyerFrigateNo match

When input is b

GCD of Two Numbers

The greatest common divisor (GCD) of two integers is the product of the integers’ common factors. Write a program that inputs two numbers and find their GCD by repeated division. For example, consider the numbers 252 and 735. find the remainder of one divided by the other.



GCD of Two Numbers

Now we calculate the remainder of the old divisor divided by the remainder found

$$\begin{array}{r} 2 \\ 252\,\overline{)\,735} \\ \underline{504} \\ 231 \end{array}$$

Repeat the process until remainder is zero

The Divisor when remainder is zero is the GCD

$$\begin{array}{r} 1 \\ 231\,\overline{)\,252} \\ \underline{231} \\ 21 \end{array} \qquad \begin{array}{r} 11 \\ 21\,\overline{)\,231} \\ \underline{21} \\ 21 \\ \underline{21} \\ 0 \end{array}$$

21 is the GCD

GCD Problem

Input	Output	Logic Involved
Two numbers	GCD of the numbers	Euclidean algorithm, binary GCD algorithm, repeated division method

Algorithm

```
Step 1: Read the numbers from the user
Step 2: Let dividend = number1 and divisor = number2
Step 3: Repeat step 4 to step 6 while remainder not equal to zero
Step 4: remainder = number1 modulus number2
Step 5: dividend = divisor
Step 6: divisor = remainder
Step 7: GCD = divisor
Step 8: print GCD
```

Implementation

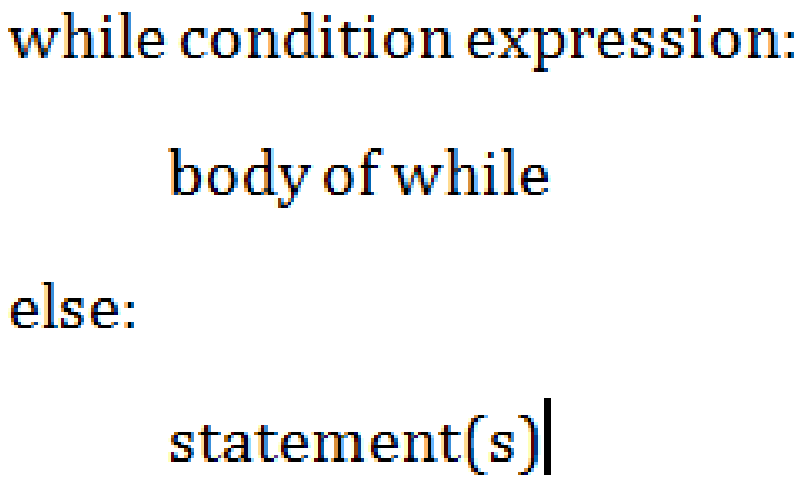
- We have to learn how to repeat statements
- In some cases the number of times to repeat a statement is known, in weather report example it is ten times we have to repeat some statements
- In some other cases the conditions are not direct as a number but as a terminating condition that may be based on I/O. In our GCD problem, the statements are to be repeated till reminder becomes zero

While loop

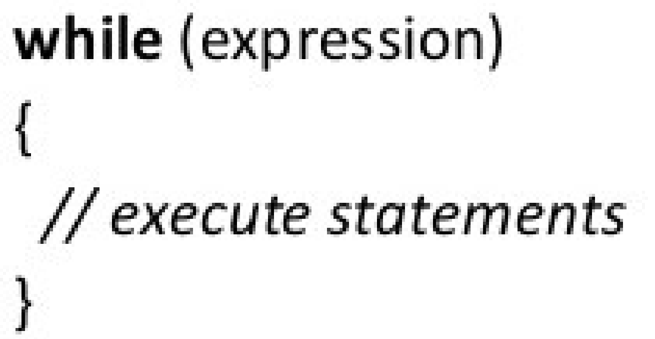
To repeat a set of statements either while a condition is met or till a condition is met

while (loop control variable < final value)   . . .
Change value of loop control variable

While statement in Python and C



```
while condition expression:

        body of while

else:

        statement(s)
```



```
while (expression)
{
  // execute statements
}
```

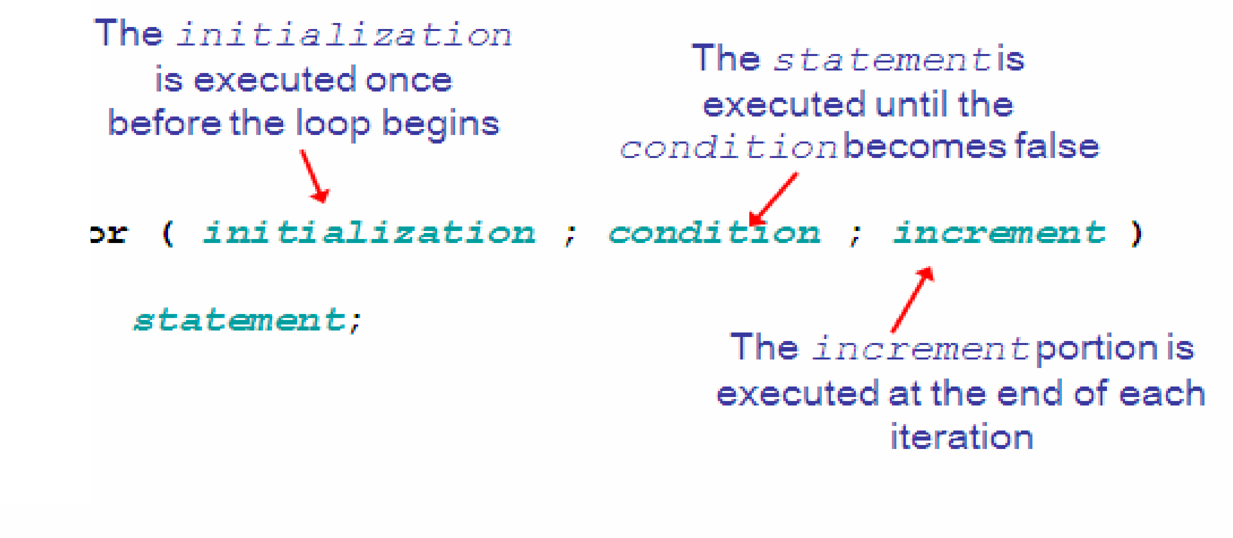
Syntax for For loop

for (loop control variable initialization; loop terminating condition; loop control variable update)

- All three components are optional
- But semicolons are mandatory

For Statement in Python and C

```
for val in sequence:
    body of for
else:
    statement(s)
```

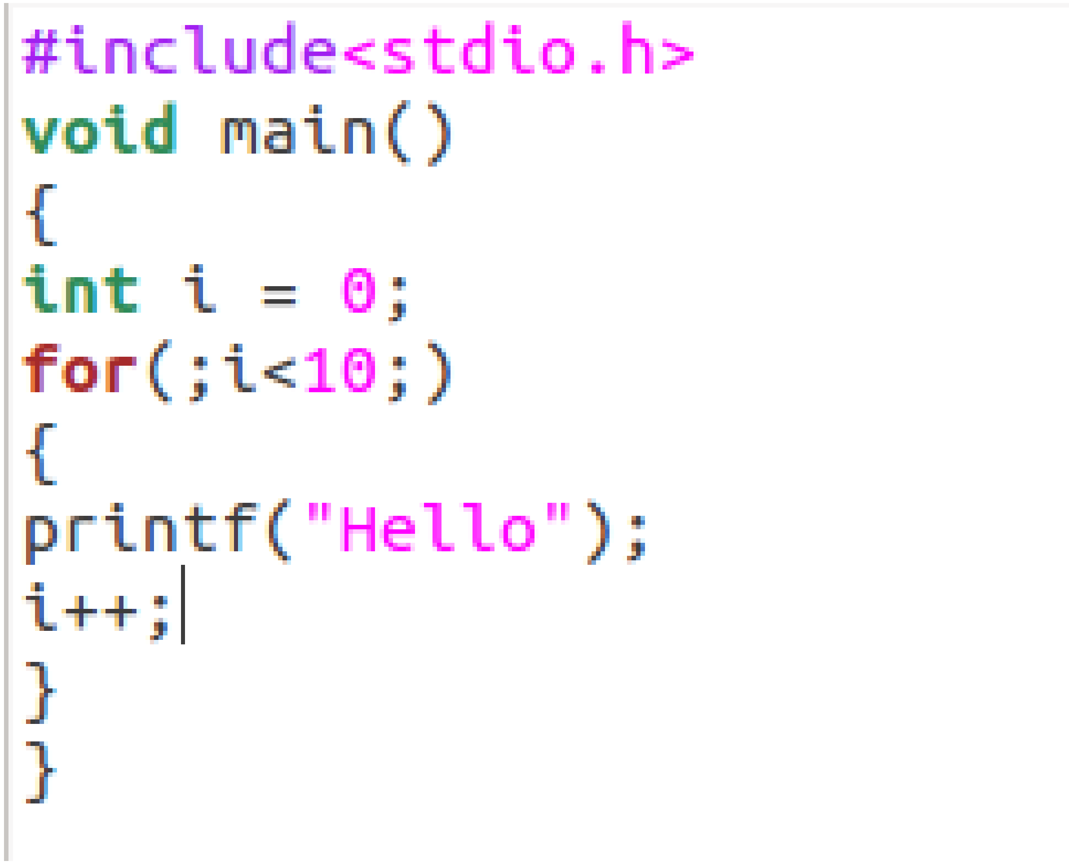


The *initialization* is executed once before the loop begins

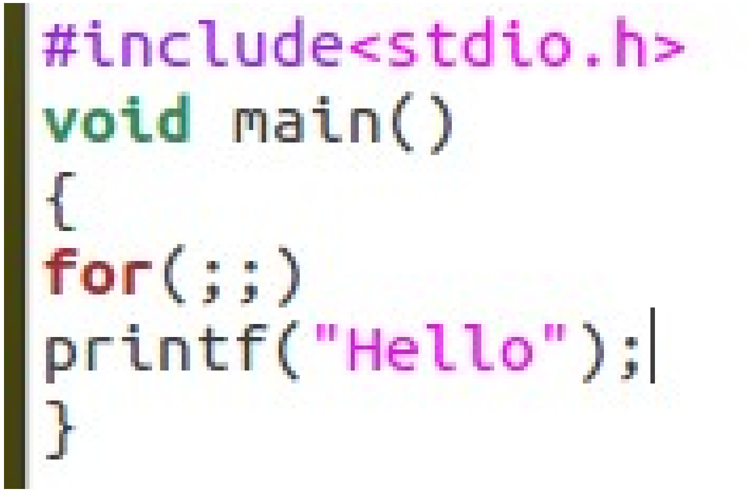
The *statement* is executed until the *condition* becomes false

```
or ( initialization ; condition ; increment )
  statement;
```

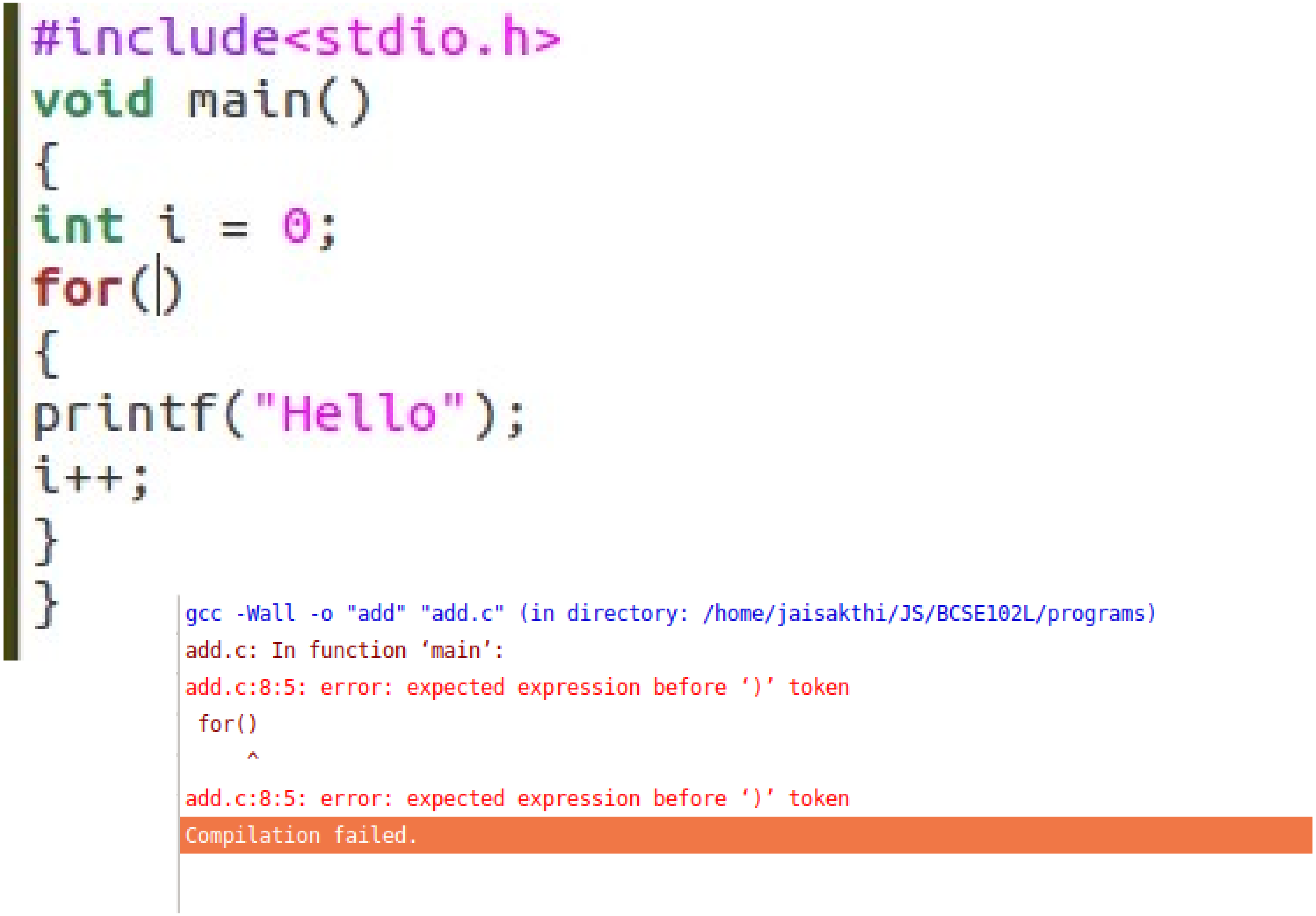
The *increment* portion is executed at the end of each iteration



```
#include<stdio.h>
void main()
{
int i = 0;
for(;i<10;)
{
printf("Hello");
i++;
}
}
```



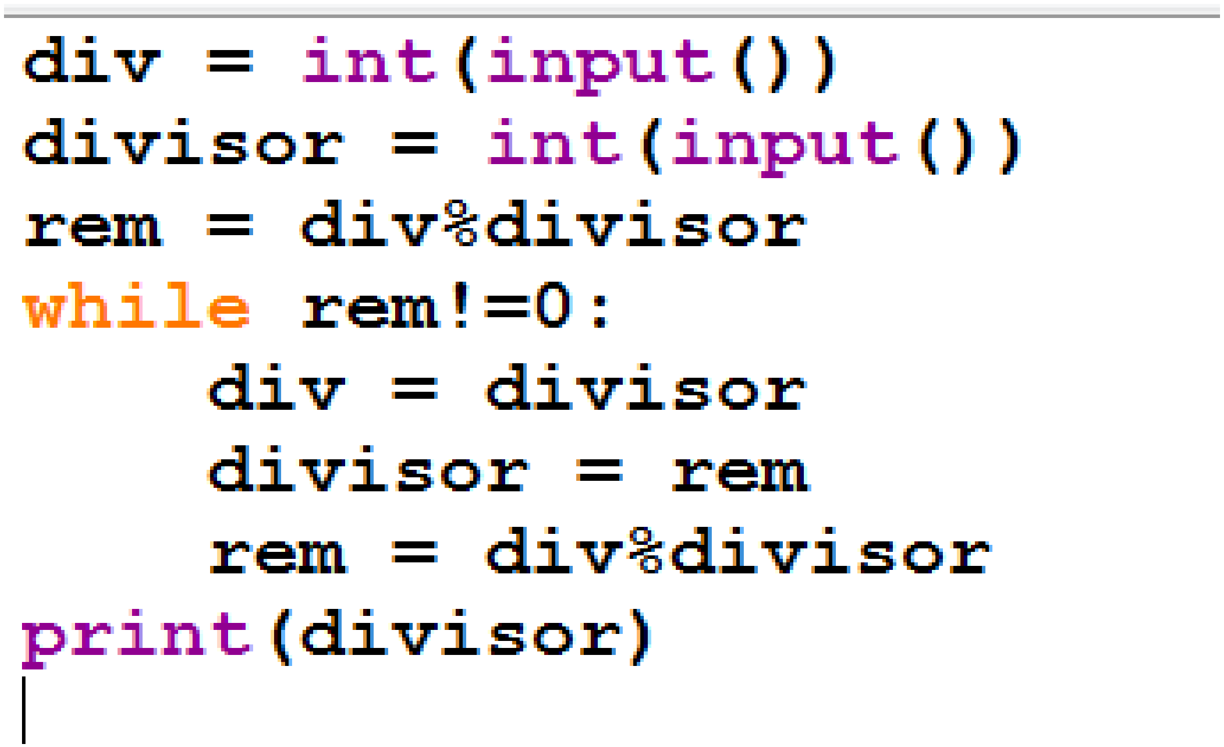
```
#include<stdio.h>
void main()
{
for(;;)
printf("Hello");
}
```



```c
#include<stdio.h>
void main()
{
int i = 0;
for()
{
printf("Hello");
i++;
}
}
```

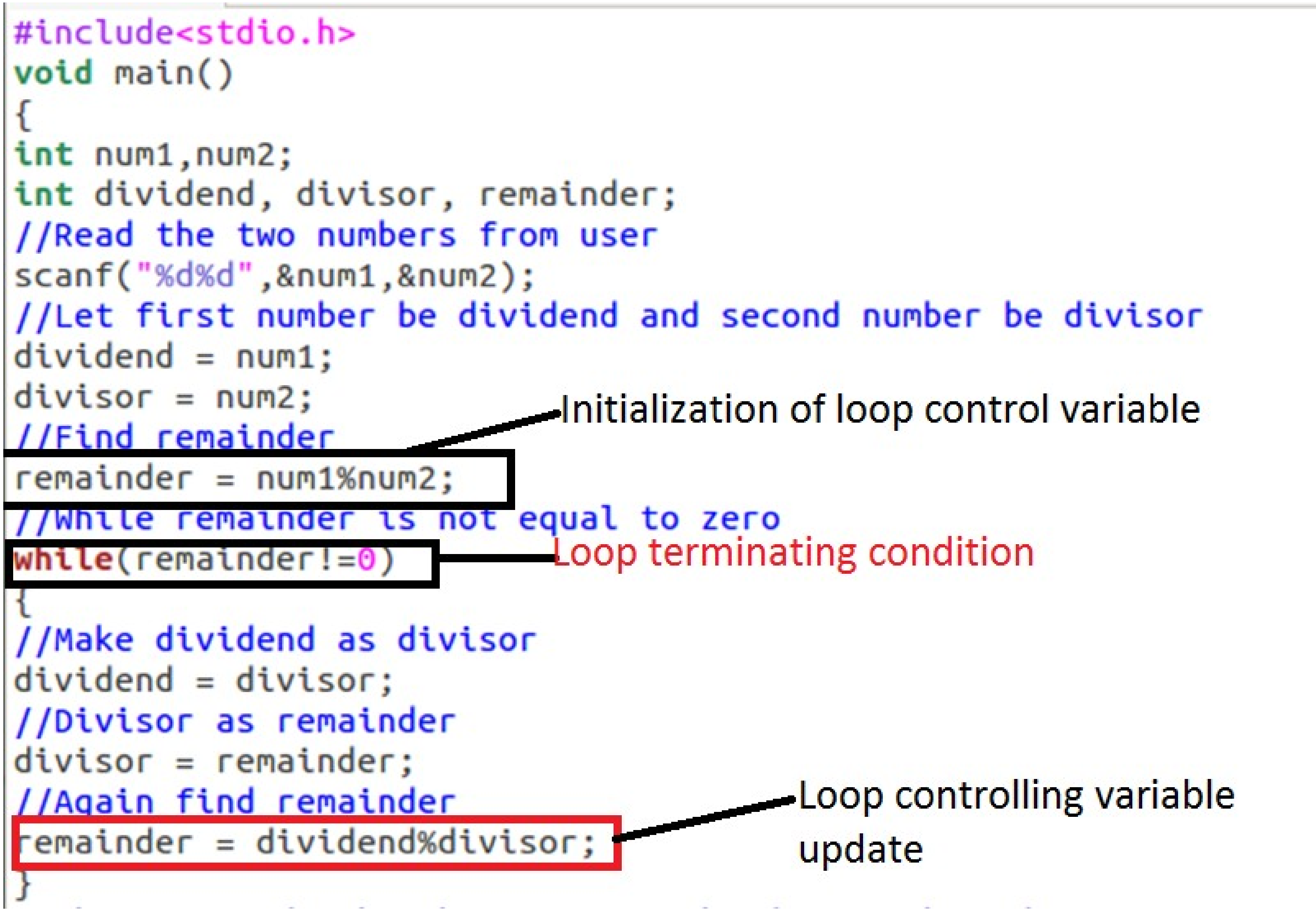
```
gcc -Wall -o "add" "add.c" (in directory: /home/jaisakthi/JS/BCSE102L/programs)
add.c: In function ‘main’:
add.c:8:5: error: expected expression before ‘)’ token
 for()
     ^
add.c:8:5: error: expected expression before ‘)’ token
Compilation failed.
```

GCD Program in Python using While



```
div = int(input())
divisor = int(input())
rem = div%divisor
while rem!=0:
    div = divisor
    divisor = rem
    rem = div%divisor
print(divisor)
```

GCD using While in C



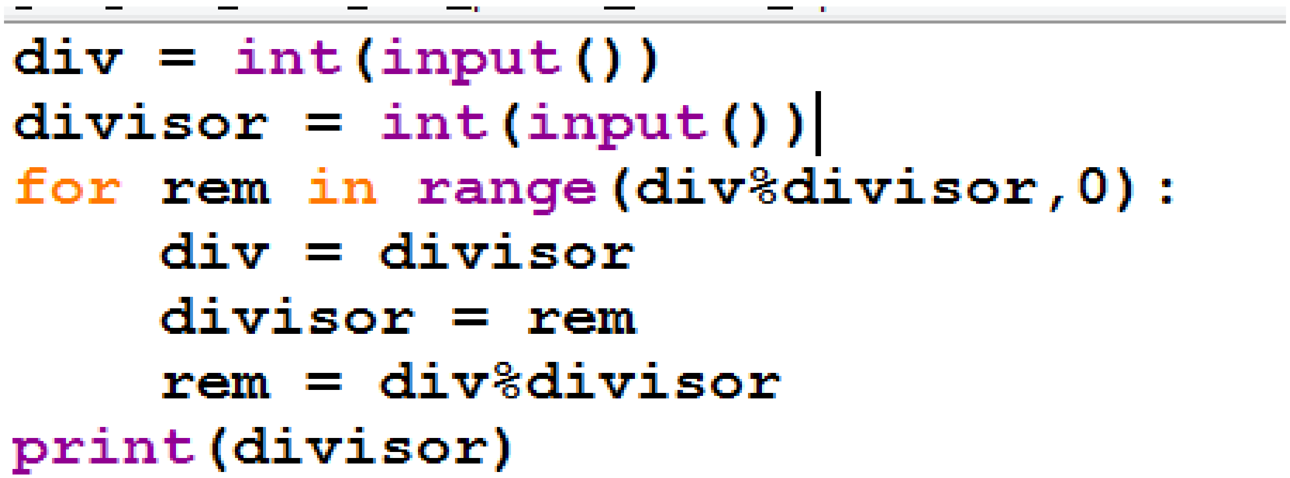
```c
#include<stdio.h>
void main()
{
int num1,num2;
int dividend, divisor, remainder;
//Read the two numbers from user
scanf("%d%d",&num1,&num2);
//Let first number be dividend and second number be divisor
dividend = num1;
divisor = num2;
//Find remainder
remainder = num1%num2;
//While remainder is not equal to zero
while(remainder!=0)
{
//Make dividend as divisor
dividend = divisor;
//Divisor as remainder
divisor = remainder;
//Again find remainder
remainder = dividend%divisor;
}
```

Initialization of loop control variable

Loop terminating condition

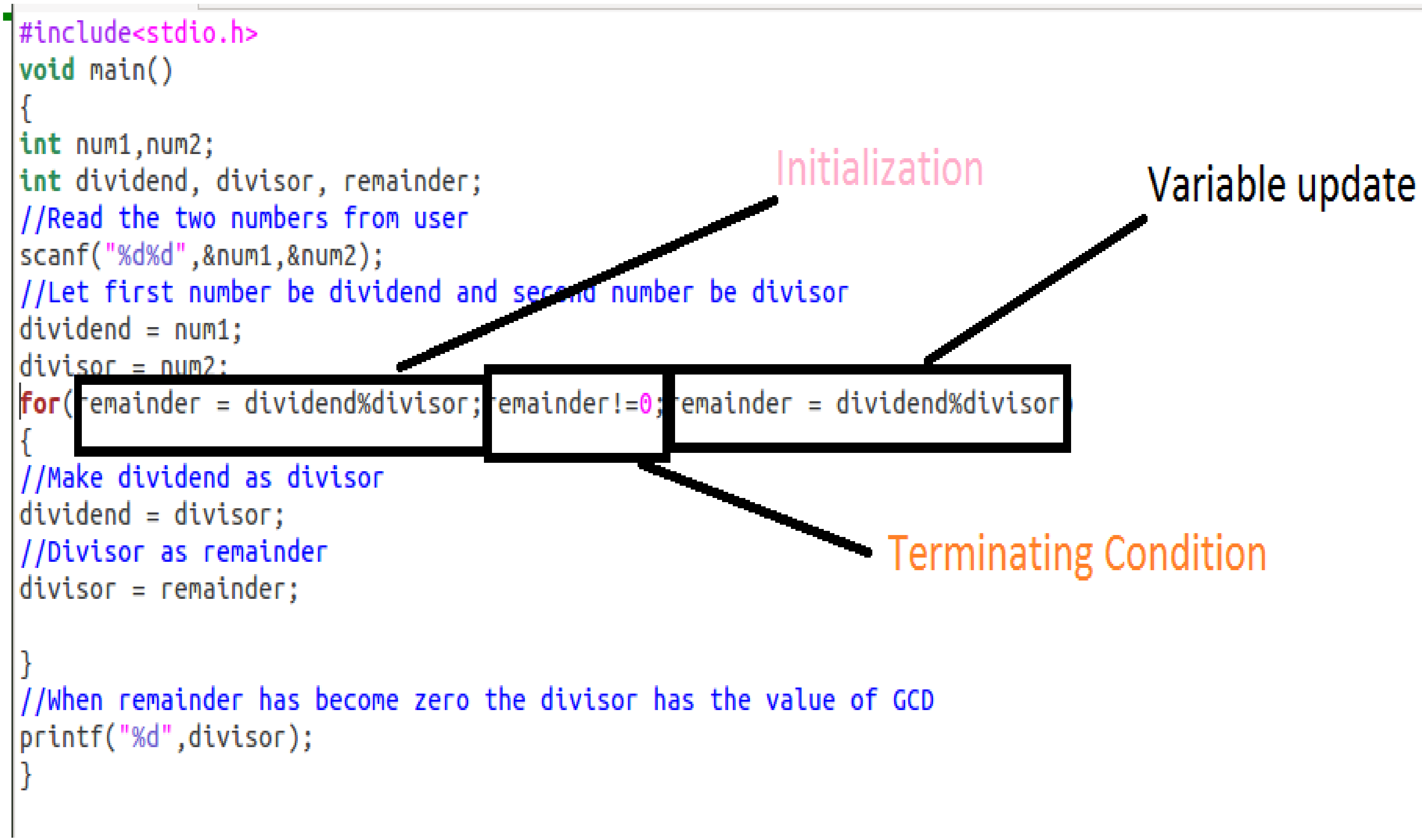
Loop controlling variable update

GCD Program in Python using For Loop



```
div = int(input())
divisor = int(input())
for rem in range(div%divisor,0):
    div = divisor
    divisor = rem
    rem = div%divisor
print(divisor)
```

GCD Program using For in C



```
#include<stdio.h>
void main()
{
int num1,num2;
int dividend, divisor, remainder;
//Read the two numbers from user
scanf("%d%d",&num1,&num2);
//Let first number be dividend and second number be divisor
dividend = num1;
divisor = num2;
for(remainder = dividend%divisor;remainder!=0;remainder = dividend%divisor)
{
//Make dividend as divisor
dividend = divisor;
//Divisor as remainder
divisor = remainder;

}
//When remainder has become zero the divisor has the value of GCD
printf("%d",divisor);
}
```

Initialization

Variable update

Terminating Condition

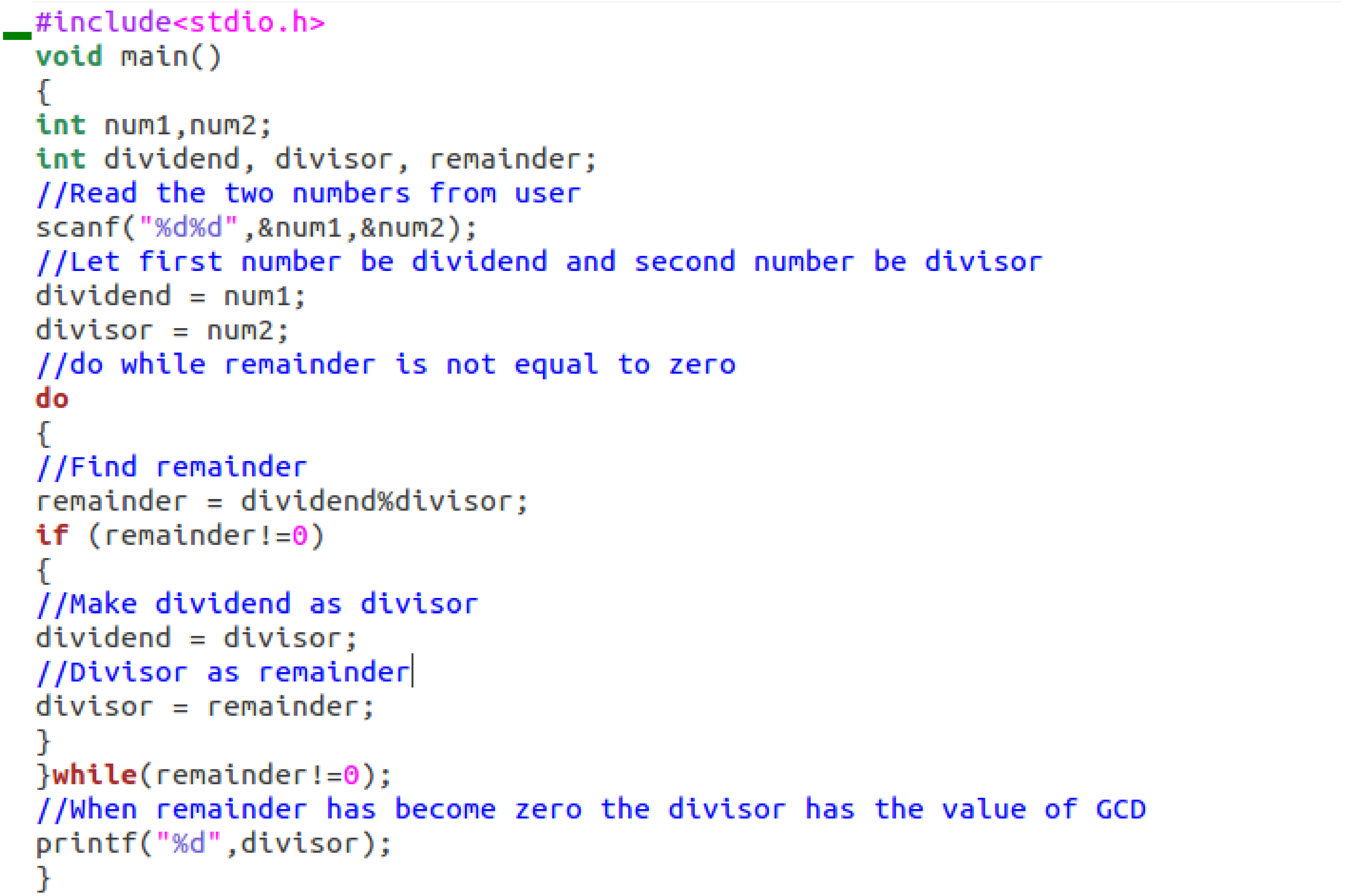
Do While loop

- Similar to a while loop, except the fact that it is guaranteed to execute at least one time
- Syntax :

```
do
{
statement(s);
} while( condition );
```

Condition is checked at the end of execution

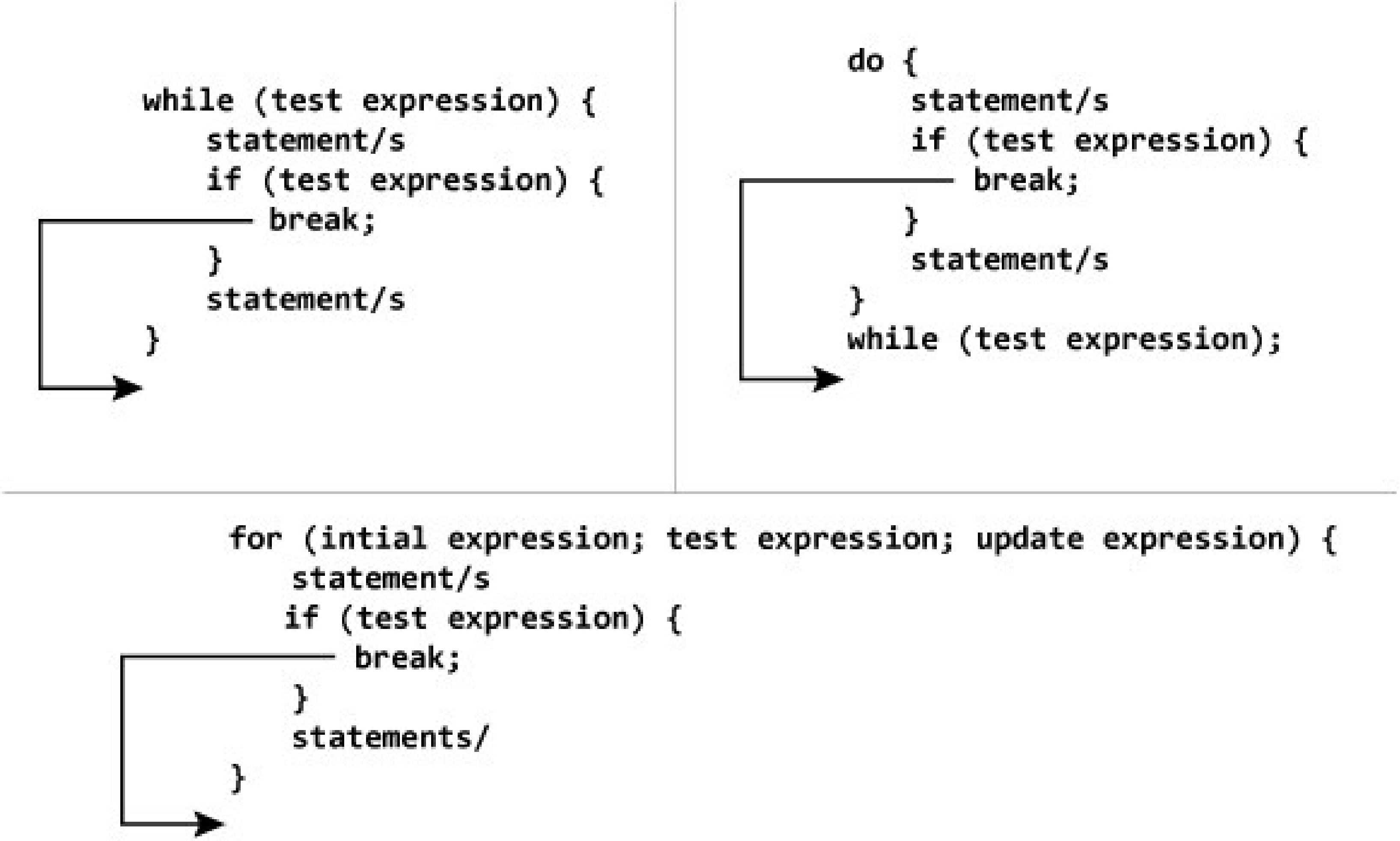
GCD Program using Do While Loop



```
#include<stdio.h>
void main()
{
int num1,num2;
int dividend, divisor, remainder;
//Read the two numbers from user
scanf("%d%d",&num1,&num2);
//Let first number be dividend and second number be divisor
dividend = num1;
divisor = num2;
//do while remainder is not equal to zero
do
{
//Find remainder
remainder = dividend%divisor;
if (remainder!=0)
{
//Make dividend as divisor
dividend = divisor;
//Divisor as remainder
divisor = remainder;
}
}while(remainder!=0);
//When remainder has become zero the divisor has the value of GCD
printf("%d",divisor);
}
```

Break and Continue Statement

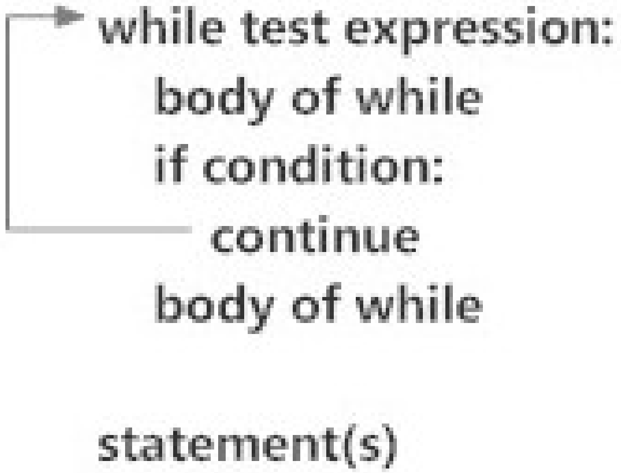
- Interrupt iterative flow of control in loops
- Break causes a loop to end
- Continue stops the current iteration and begin the next iteration



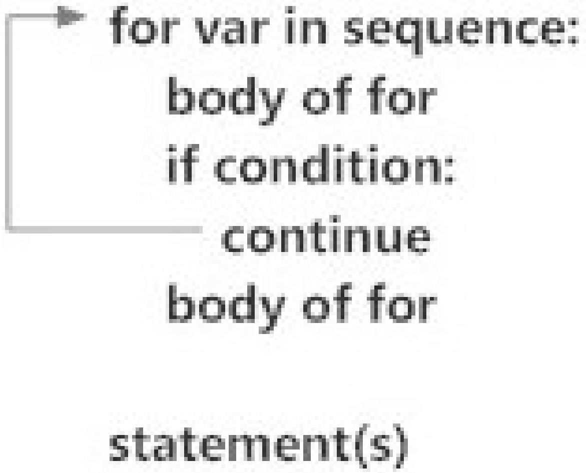
```
while (test expression) {
    statement/s
    if (test expression) {
        break;
    }
    statement/s
}
```

```
do {
    statement/s
    if (test expression) {
        break;
    }
    statement/s
}
while (test expression);
```

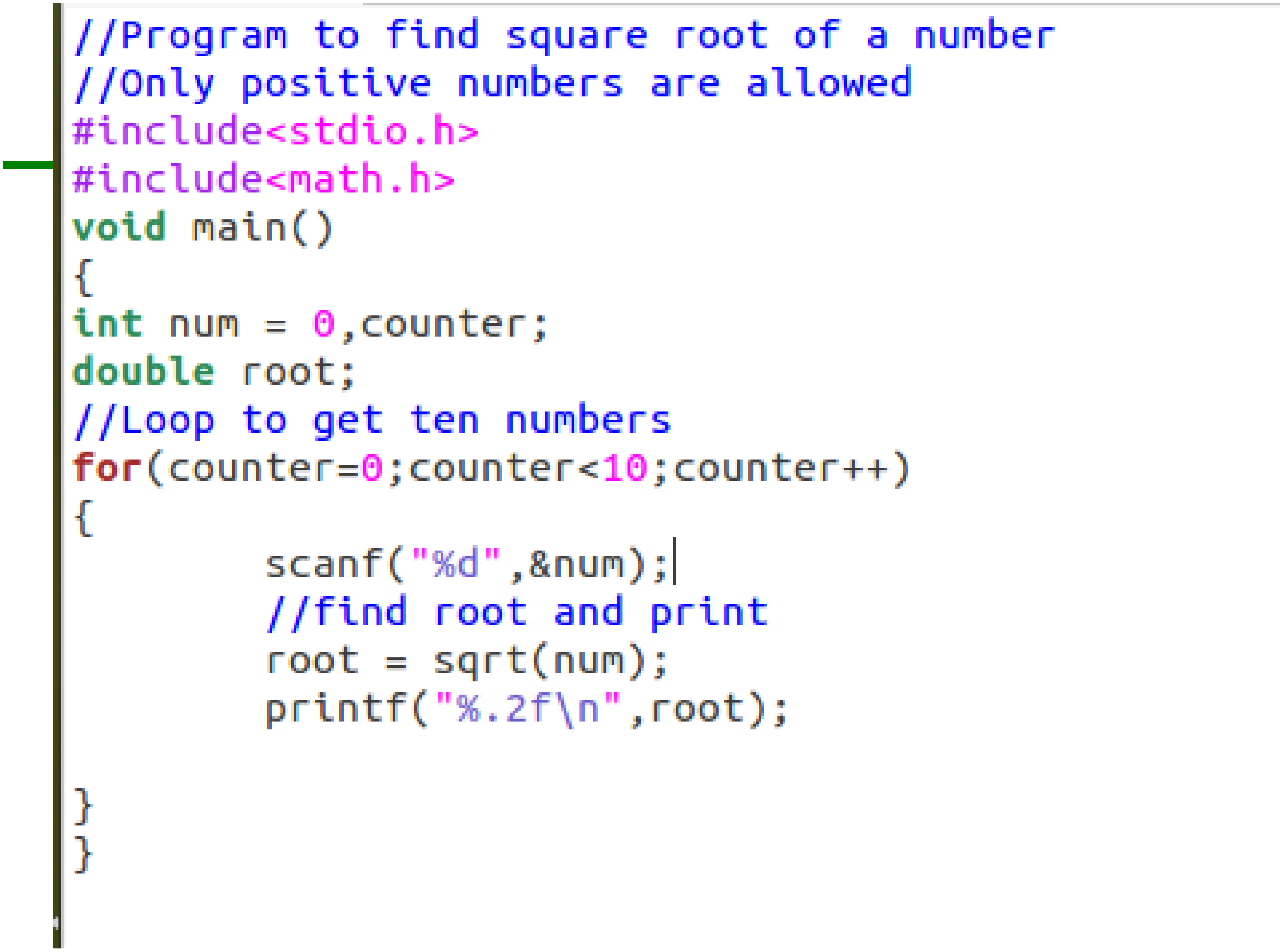
```
for (intial expression; test expression; update expression) {
    statement/s
   if (test expression) {
        break;
    }
    statements/
}
```



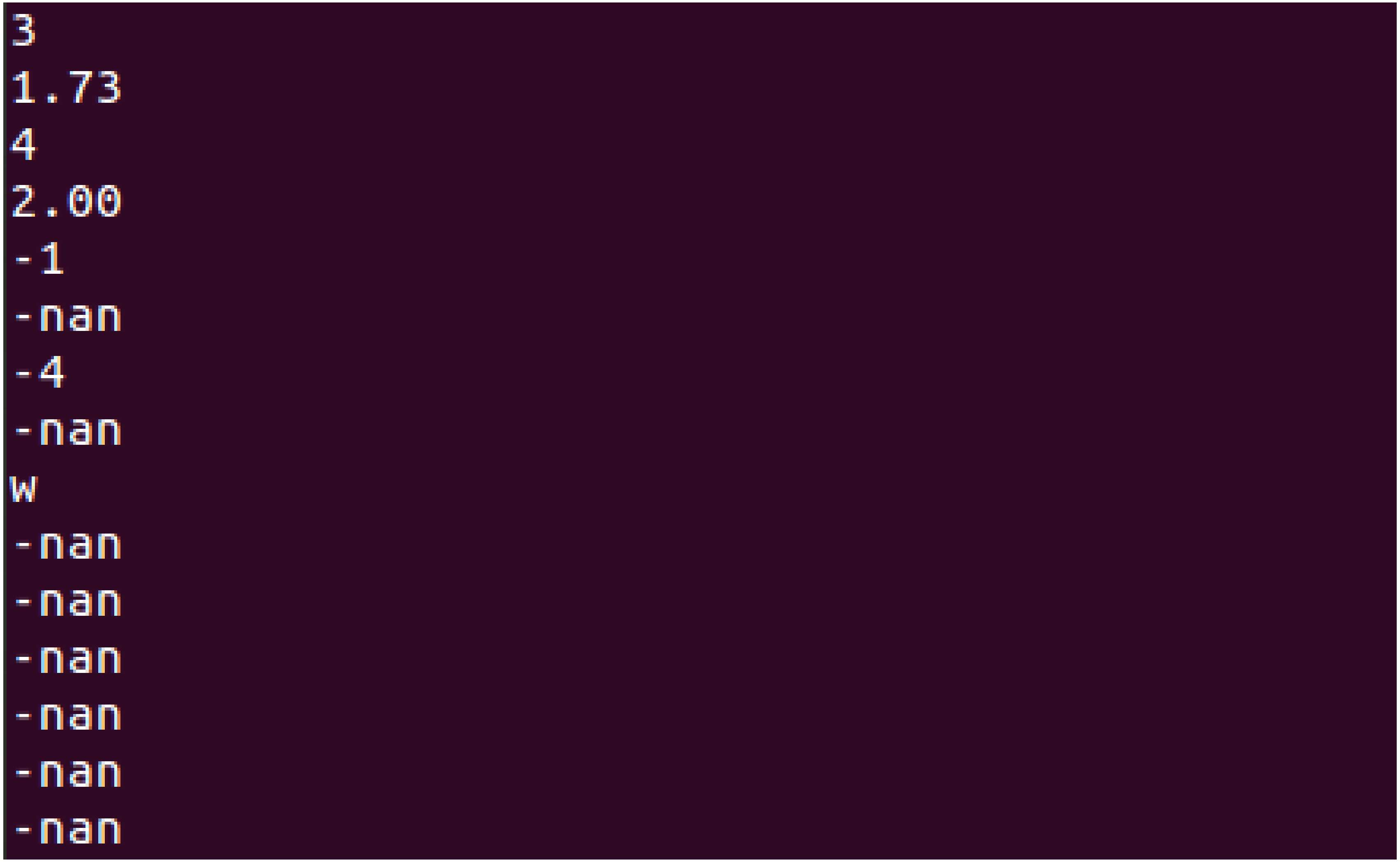
```
while test expression:
    body of while
    if condition:
        continue
    body of while

statement(s)
```



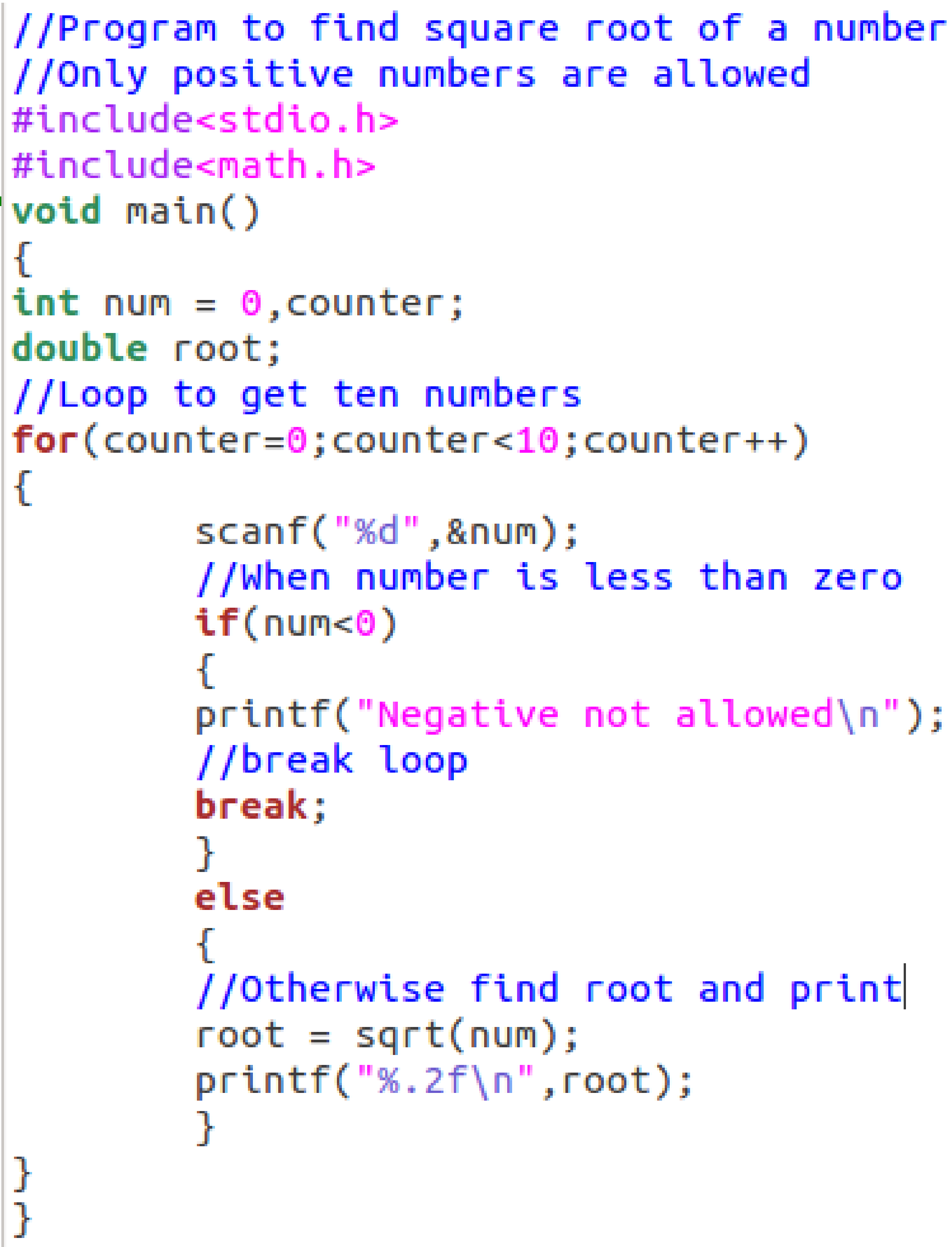
```
for var in sequence:
    body of for
    if condition:
        continue
    body of for

statement(s)
```



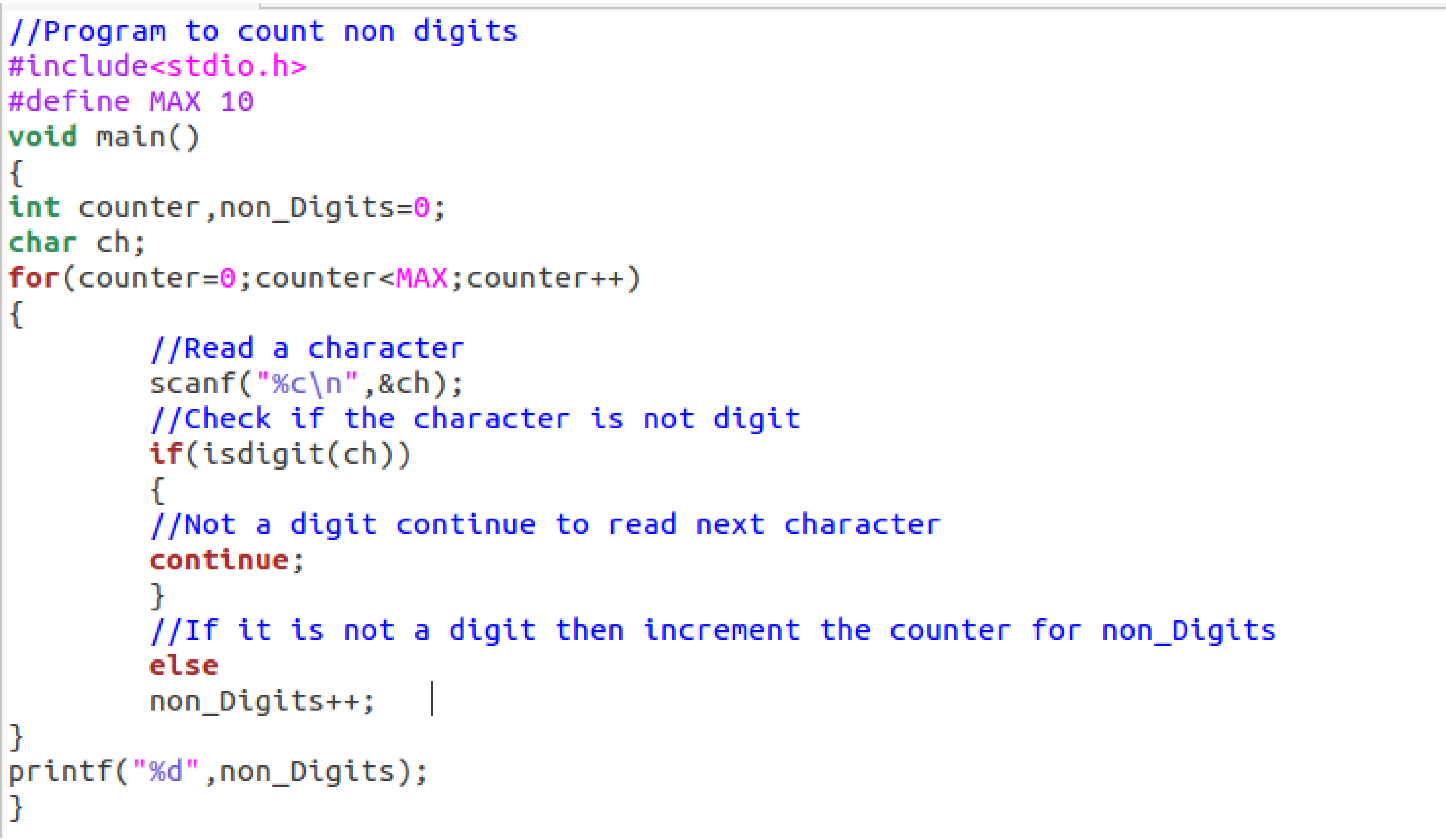
```
//Program to find square root of a number
//Only positive numbers are allowed
#include<stdio.h>
#include<math.h>
void main()
{
int num = 0,counter;
double root;
//Loop to get ten numbers
for(counter=0;counter<10;counter++)
{
        scanf("%d",&num);
        //find root and print
        root = sqrt(num);
        printf("%.2f\n",root);


}
}
```



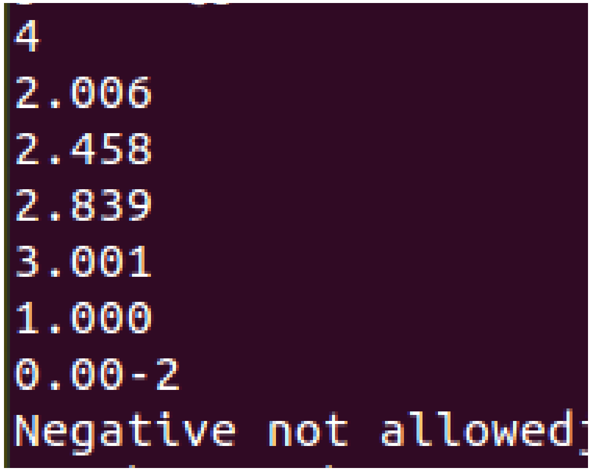
```
3
1.73
4
2.00
-1
-nan
-4
-nan
w
-nan
-nan
-nan
-nan
-nan
-nan
```



```
//Program to find square root of a number
//Only positive numbers are allowed
#include<stdio.h>
#include<math.h>
void main()
{
int num = 0,counter;
double root;
//Loop to get ten numbers
for(counter=0;counter<10;counter++)
{
        scanf("%d",&num);
        //When number is less than zero
        if(num<0)
        {
        printf("Negative not allowed\n");
        //break loop
        break;
        }
        else
        {
        //Otherwise find root and print
        root = sqrt(num);
        printf("%.2f\n",root);
        }
}
}
```



```
//Program to count non digits
#include<stdio.h>
#define MAX 10
void main()
{
int counter,non_Digits=0;
char ch;
for(counter=0;counter<MAX;counter++)
{
        //Read a character
        scanf("%c\n",&ch);
        //Check if the character is not digit
        if(isdigit(ch))
        {
        //Not a digit continue to read next character
        continue;
        }
        //If it is not a digit then increment the counter for non_Digits
        else
        non_Digits++;
}
printf("%d",non_Digits);
}
```



```
4
2.006
2.458
2.839
3.001
1.000
0.00-2
Negative not allowed;
```