Attendance record

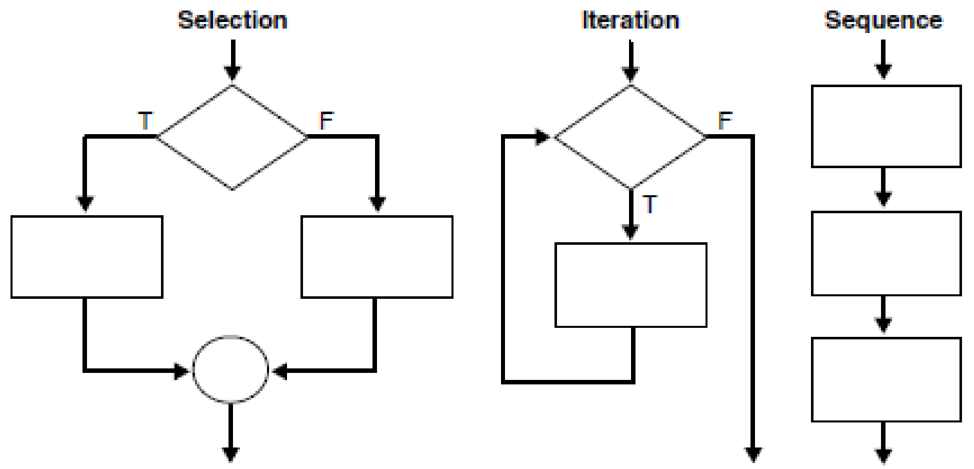
- To check what devices students are using to join class
- To check attendance

https://forms.gle/CPUZBoHBM2NnJU2CA

Start class: **3:05pm**

DATA STRUCTURE & PROGRAMMING I

Chapter 3- Control structures (decision making and loop)



Prepared by: Mr. VALY Dona and Mr. NOU Sotheany

Modified and Lectured by: Mr. BOU Channa

Lecture overview

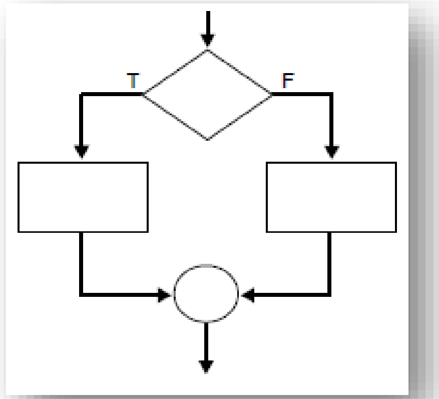
Overall lectures

1. Introduction to algorithm
2. Basic data types and statements
3. **Control structures (decision making and loop)**
4. Array
5. Data structure
6. Sub-programs

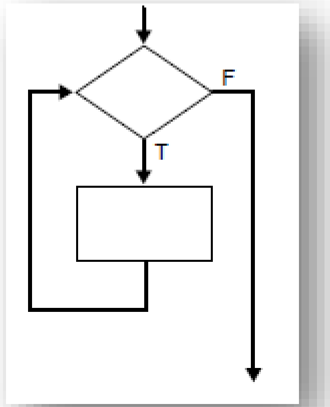
Objective and Outcome of the lesson

Upon completion of the lesson, students will be able to:

- Control flow of the program
- Write a program more logically using various decision making statements
  - `if`
  - `else if`
  - `else`



- Write an efficient program using loops
  - `for`
  - `while`
  - `do … while`



Overview

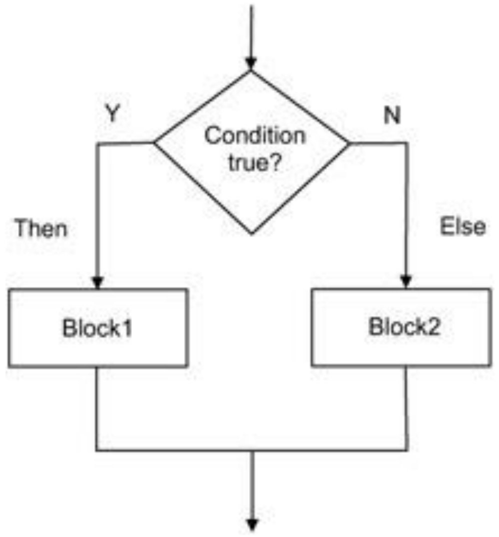
Outline

- Introduction to control structure
- Control structure for decision making
    - `if, else if, else`
- Introduction to loop
- Types of loops
    - `for`
    - `while`
    - `do … while`

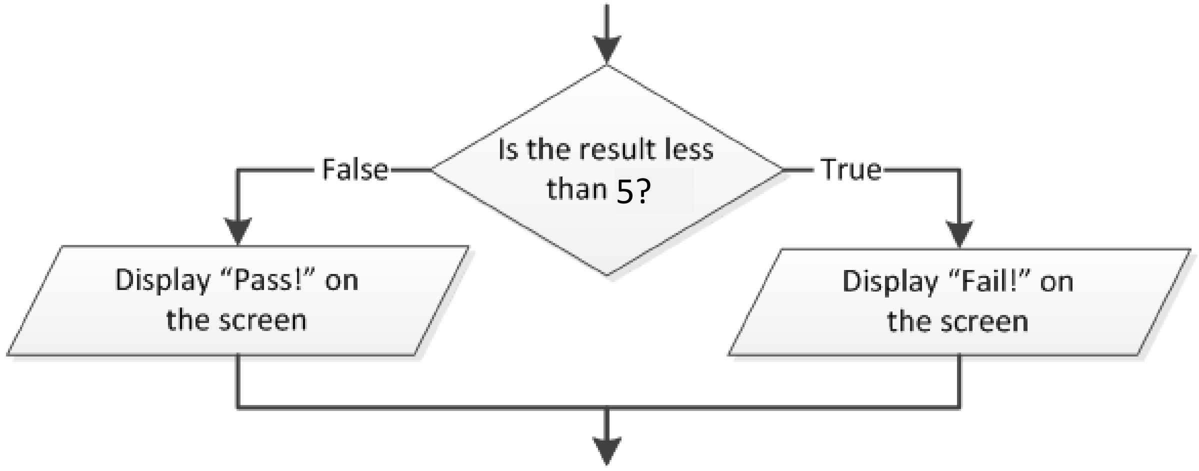
Introduction

What is control structure?

- It is the element of language that determines which block of statements should be executed
- Control structures:
  - Decision making
  - Loop



Decision making


Is the result less than 5?
False
True
Display "Pass!" on the screen
Display "Fail!" on the screen

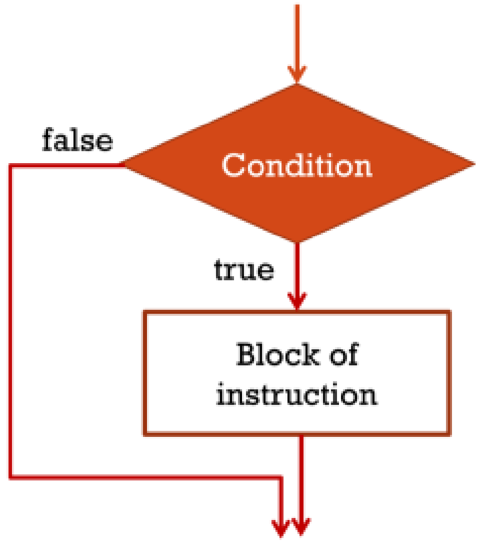
Example of decision making

Decision Making

Condition IF

- It execute instruction in some condition
- Syntax

```
if (condition) then
        block of instruction
end if
```



- *condition* is relational condition which returns *true* or *false*
  - Ex: a>b, a==b, a<=b (a and b should be defined and contain some values before)
- The block of instruction is executed only if *condition* return *true*
- If the *condition* return *false*, nothing happen

Decision Making

Examples

```
if (3<2) then
      write(“Hello\n”)
end if
write(“Hello 2”)
```

Example 1

```
a ← 2
b ← 3
if (a<b) then
      write(“Hi,”)
      write(“Welcome back!\n”)
end if
write(“Hello”)
```

Example 2

Output:

```
Hello 2
```

Output:

```
Hi, Welcome back!
Hello
```

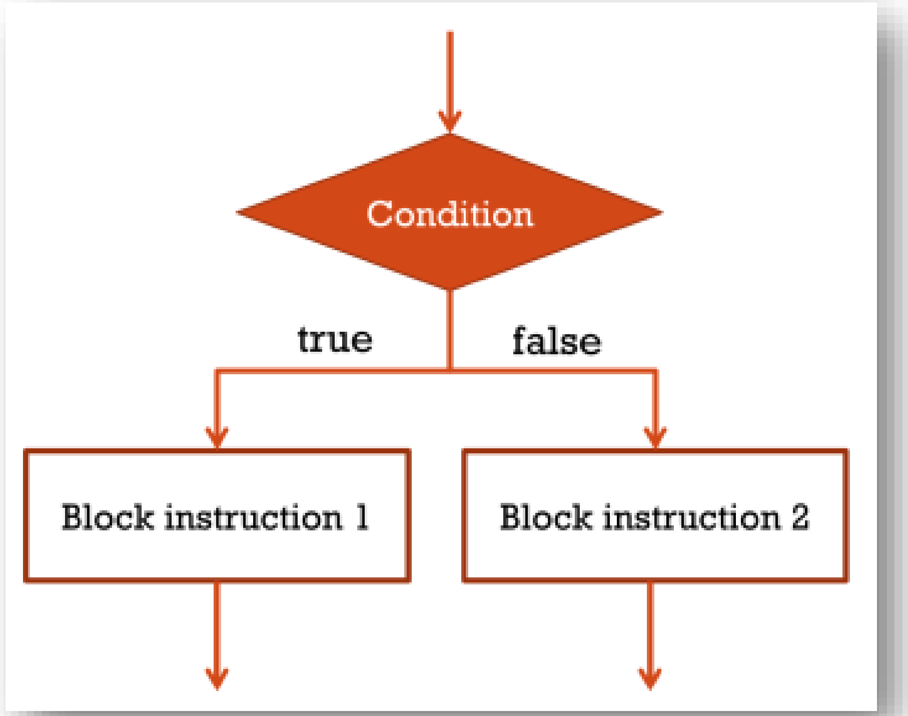
Decision Making

Condition IF and ELSE

- Syntax

```
if (condition) then
        block 1 instructions
else
        block 2 instructions
end if
```

  - When *condition* return *true*, block 1 is executed
  - When *condition* return *false*, block 2 is executed



Decision Making

Examples

```
a ← 9
if (a<9) then
      write(“Condition return true”)
else
      write(“Condition return false”)
end if
write(“Hello 2”)
```

Example 3

Output:

```
Condition return false
Hello 2
```

```
a ← 10
b ← 50
if (a<b) then
      write(“Condition return true”)
else
      write(“Condition return false”)
end if
write(“Hello 2”)
```

Example 4

Output:

```
Condition return true
Hello 2
```

Decision Making

Example

```
Var a, b: Integer
Begin
    read(a,b)
    if (a>b) then
        write ("The greater value is:", a)
    else
        write("The smaller value is:", b)
    end if
End
```

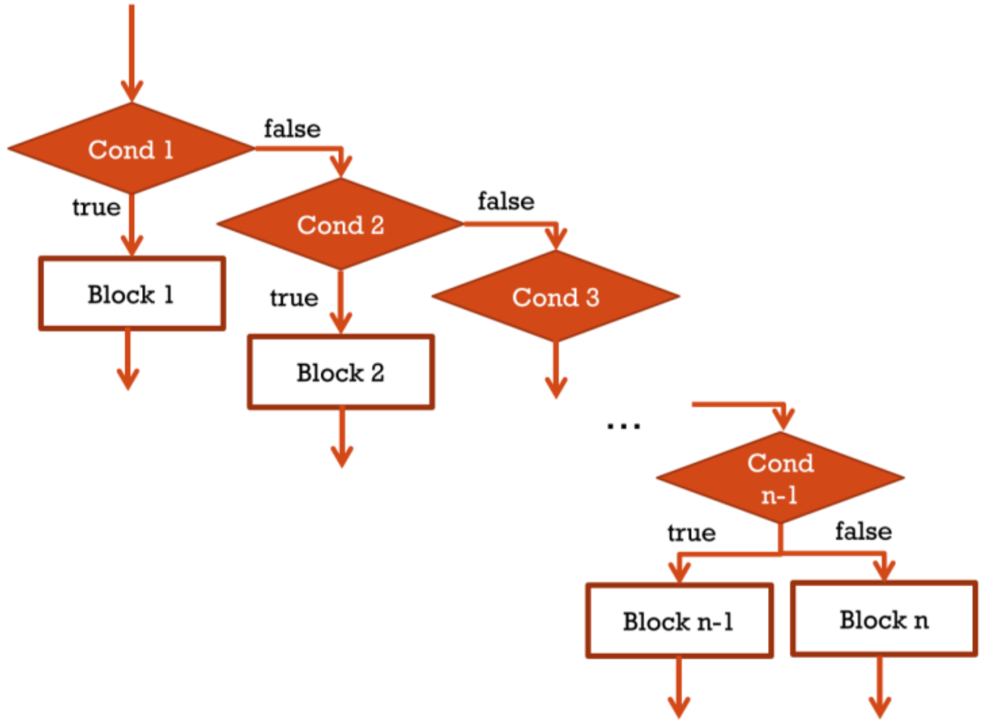
Example 5: Get two values from a user then check the bigger and the smaller value

Decision Making

Condition IF with ELSE IF

- Syntax

```
if (condition 1) then
      block of instructions 1
else if (condition 2) then
      block of instructions 2
else if (condition 3) then
      block of instructions 3
.
.
.
else if (condition n-1) then
      block instruction n-1
else
      block of instruction n
end if
```



Decision Making

Example

```
Var x: Integer
Begin
    read(x)
    if (x >= 0) then
        write(“x is positive number”)
    else if (x < 0) then
        write(“x is negative number”)
    end if
End
```

```
Var x: Integer
Begin
    read(x)
    if (x < 0) then
        write(“x is negative number”)
    else
        write(“x is positive number”)
    end if
End
```

Example 6: Get a number from a user then check whether it is positive or negative number

Decision Making

Example

```
Var x: Integer
Begin
    read(x)
    if (x == 100) then
        write(“x is equal to 100”)
    else if (x > 100) then
        write(“x is greater than 100”)
    else if (x < 100) then
        write(“x is less than 10”)
    end if
End
```

Example 7: Get a number from a user then check whether if is equal to 100, more than 100 or less than 100

Decision Making

Example

```
Var x: Integer
Begin
    read(x)
    if (x >= 100) then
        write("x is greater than or equal 100")
    else if (x > 50) then
        write("x is greater than 50 but less than 100")
    else
        write("x less than or equal 50")
    end if
End
```

Example 8: Get a number from a user then check if the number is greater than or equal 100, greater than 50 but less than 100, the rest condition.

Decision Making

Compare two algorithms below

```
Var x: Integer
Begin
    read(x)
    if (x>10) then
        write(“x>10”)
    end if
    if (3<x<=10) then
        write(“3<x<=10”)
    end if
    if (0<x<=7) then
        write(“0<x<=7”)
    end if
End
```

Algorithm 1

```
Var x: Integer
Begin
    read(x)
    if (x>10) then
        write(“x>10”)
    else if (x>5) then
        write(“5<x<=10”)
    else if (x>0) then
        write(“0<x<=5”)
    end if
End
```

Algorithm 2

Nested condition

Remark

- Nested condition is implemented if more than 2 possible conditions are needed

```
Var x: Integer
Begin
    read(x)
    if (x<0) then           //Condition 1
        write("It is a negative number.")
    else                    //Condition 2
        if (x==0) then         //Sub-condition 2.1
            write("It is zero.")
        else                   //Sub-condition 2.2
            write("It is a positive number.")
        end if
    end if
    write("Quitting the program ...")
End
```

Q & A

Practices and Discussion

Step 1: Divide group (5mn)

How to divide group

- Five groups (G1 – G5)
- Students count from 1 to 5,
- Those got number 1, he/she belongs to group 1 (G1)
- Similarly, G2, G3, G4, and G5 for those got number 2, 3, 4 and 5, respectively.

Step 2: Work in group (15mn)

Instruction

- Each member choose an exercise and solve it by writing an algorithm
- Each member explain their solution for each exercise to his/her team

Step 3: Sharing to class (10mn)

Instruction

- Each group explains to class for each exercise
  - A member for a group is randomly selected

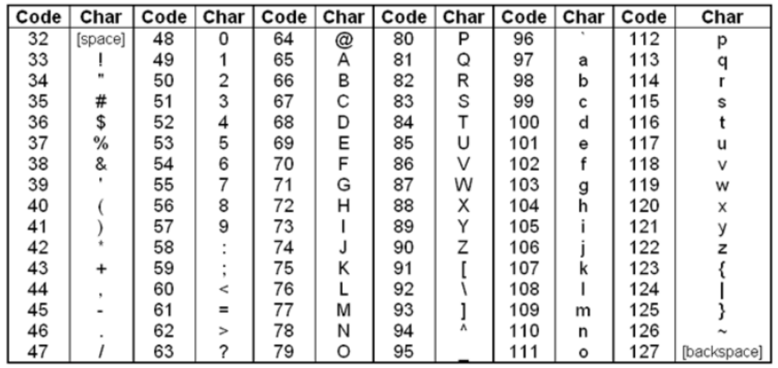
Practices and Discussion

Break 10mn
Start: 8:45pm

Exercises

1. Write an algorithm to tell the grade of a score. The user input a score then program displays grade of the score using the grading method below:
   - Greater than or equal 90, grade “A”
   - Greater than or equal 80, grade “B”
   - Greater than or equal 70, grade “C”
   - Greater than or equal 60, grade “D”
   - Otherwise, grade “F”

Table 1: ASCII Code Table



Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char
32	[space]	48	0	64	@	80	P	96	`	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[	107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	|
45	-	61	=	77	M	93	]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o	127	[backspace]

2. Write an algorithm to find the biggest number between 5 numbers entered by a user.

3. Write an algorithm to ask for an input character from a user and tell if that character is a number, an uppercase letter, or an lowercase letter. If not, shower a message “That is not a number nor a letter”. **Hint**: Convert a given character to a number then use ASCII code to check. E.g: ASCII code from 48 to 57, it is a number (0-9). (See Table 1 for ASCII Code)

4. Write an algorithm which requests a value of year, of month, day and tell if it is a valid date.

Practices and Discussion

Exercises

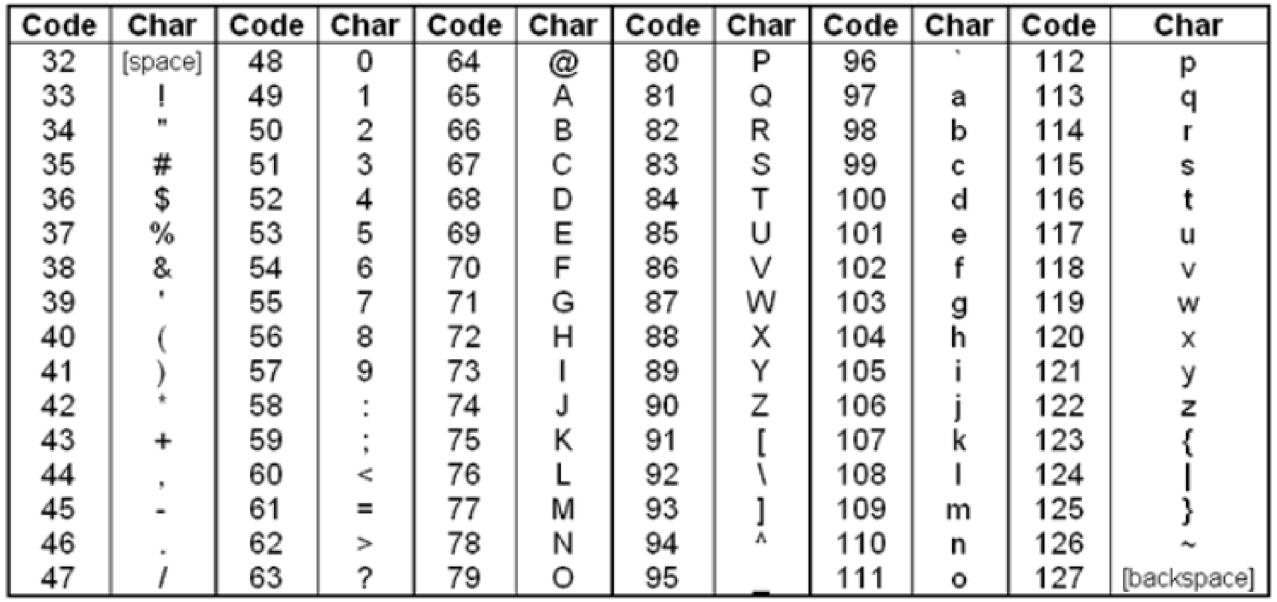
1. Write an algorithm to tell the grade of a score. The user input a score then program displays grade of the score using the grading method below:
   - When score is greater than or equal 90, then display **You got grade "A"**
   - When score is greater than or equal 80, then display **You got grade "B"**
   - When score is greater than or equal 70, then display **You got grade "C"**
   - When score is greater than or equal 60, then display **You got grade "D"**
   - Otherwise, display **You got grade "F"**

Practices and Discussion

Exercises

2. Write an algorithm to ask for an input character from a user and tell if that character is a number, an uppercase letter, or an lowercase letter. If not, show this message "It is not a number nor a letter".

Table 1: ASCII Code Table



Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char
32	[space]	48	0	64	@	80	P	96	`	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[	107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	|
45	-	61	=	77	M	93	]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o	127	[backspace]

Hint: Convert the given character to a number then use ASCII code to check.

E.g: ASCII code from 48 to 57, it is a number (0-9).

(See Table 1 for ASCII Code)

Practices and Discussion

Exercises

3. Write an algorithm to find the minimum number between 7 numbers entered by a user.

4. Write an algorithm to ask a user for year, month, and day (3 integer variables). Then tell if it is a valid date.

TD3

Exercises

1. Write an algorithm to check whether a number entered by a user is an even or odd number.
2. Write an algorithm to check if a number entered by a user is positive or negative number.
3. Write an algorithm to find root of the quadratic equation $ax^2+bx+c=0$. Ask a user to inputs the coefficient a, b and c. Find delta, find x1 and x2 based on delta value. Then display the roots.
4. Write an algorithm to ask a user for 8 numbers. Find the max number among them. Display max number on screen.

TP

Exercises

1. Write a C program to find the minimum number between 7 numbers entered by a user.

2. Write a C program to solve the quadratic equation $ax^2+bx+c=0$. Ask a user to inputs the coefficient a, b and c then display the roots.

3. Write a C program to ask a user for year, month, and day (3 integer variables). Then tell if it is a valid date.

```
Var y,m,d: Integer
Begin
     read(y, m, d)
     if(y>=0) then
          if(m==1 OR m==3 OR m==5 OR m==7 OR m==8 OR m==10 or m==12) then
              if(d>=0 AND d<=31) then
                     write(“VALID DATE)
              else
                     write(“INVALID)”)
              end if
          else if (m==4 OR m==6 …..)  //month with day being <=30
              if(d>=0 AND d<=30) then
                     write(“VALID DATE”)
              else
                     write(“INVALID”)
              end if
          else if (m==2) then
              if(m%4==0) //leap year
                     if(d>=0 AND d<=29) then
                         write(“VALID DATE”)
                     else
                         write(“INVALID”
                     end if
             else
                     …..// d>=0 AND d<=28
             end if
          end if
     End if
End
```

Solution

Exercise 1:

```
Var score: Integer
Begin
    write(“Enter your score to identify your grade: ”)
    read(score)
    if (score>=90) then
        write(“You got grade A.”)
    else if (score>=80) then
        write(“You got grade B.”)
    else if (score>=70) then
        write(“You got grade C.”)
    else if (score>=60) then
        write(“You got grade D.”)
    else
        write(“You got grade F.”)
    end if
    write(“Quitting the program ...”)
End
```

Solution

Exercise 2:

```
Var n1,n2,n3: Integer
Begin
    write("Enter three integer numbers: ")
    read(n1,n2,n3)
    if (n1>=n2 AND n1>=n3) then
        write(n1," is the biggest number.")
    else if (n2>=n1 AND n2>=n3) then
         write(n2," is the biggest number.")
    else if (n3>=n1 AND n3>=n1) then
         write(n3," is the biggest number.")
    end if
    write("Quitting the program ...")
End
```

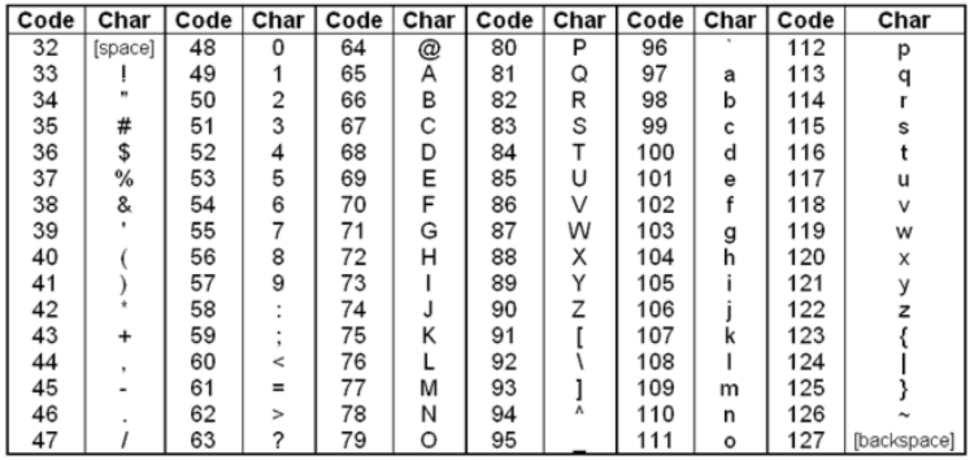
```
Var n1,n2,n3, max: Integer
Begin
    write("Enter three integer numbers: ")
    read(n1,n2,n3)
    max ← n1

    if (max<n2) then
        max ← n2
    end if
    if (max<n3) then
        max ← n3
    end if

    write(max," is the biggest number.")
    write("Quitting the program ...")
End
```

Solution

Exercise 3:

```
Var ch: Integer
Var n: Integer
Begin
    write(“Enter a character: ”)
    read(ch)
    n ← ord(ch)
    if (n>=48 AND n<=57) then
        write(“It is a number.”)
    else if (n>=65 AND n<=90) then
        write(“It is an uppercase letter.”)
    else if (n>=97 AND n<=122) then
        write(“It is a lowercase letter.”)
    else
        write(“That is not a number or a letter.”)
    end if
    write(“Quitting the program ...”)
End
```

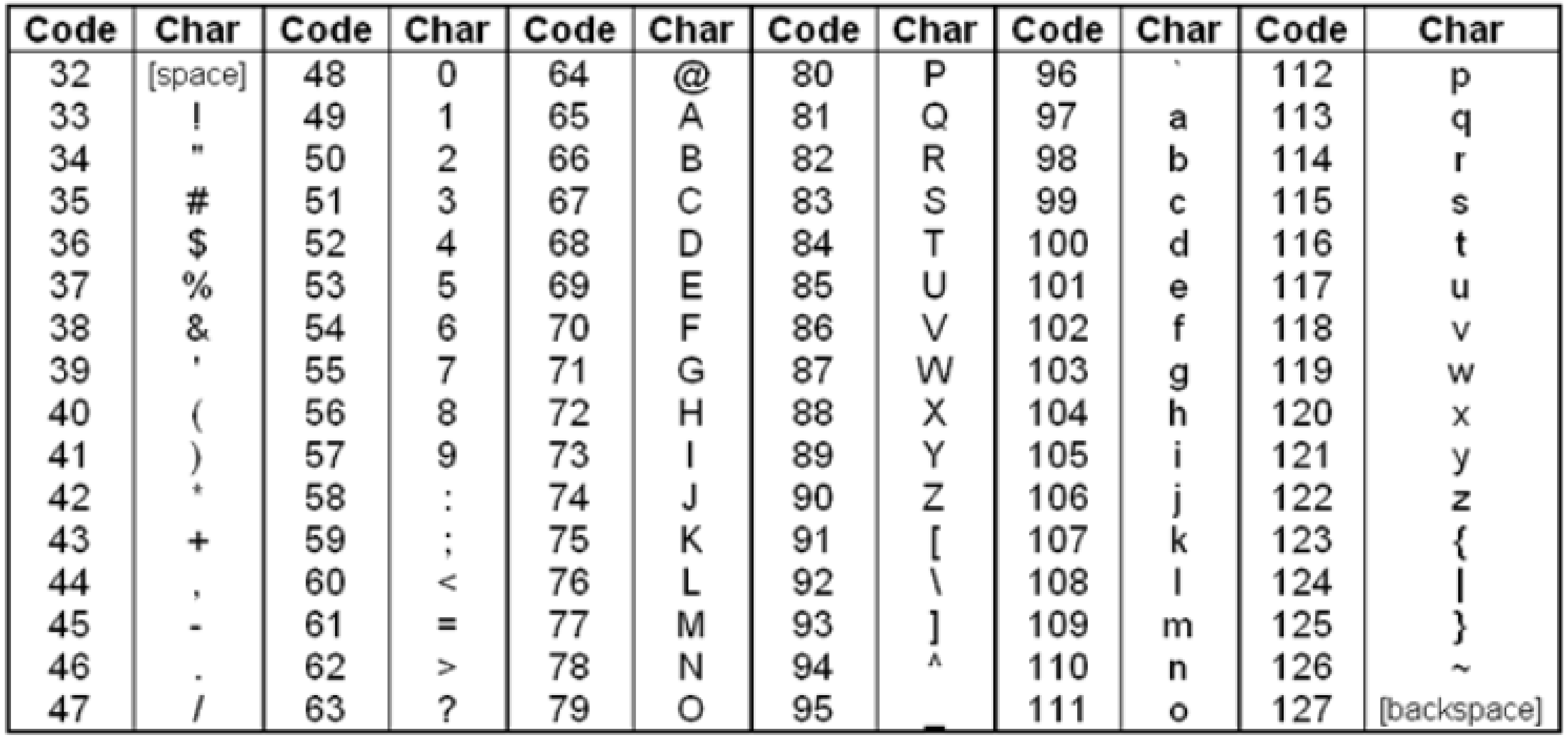
ASCII Code Table



Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char
32	[space]	48	0	64	@	80	P	96	`	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[	107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	|
45	-	61	=	77	M	93	]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o	127	[backspace]

Solution

Exercise 3:



Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char
32	[space]	48	0	64	@	80	P	96	`	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[	107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	|
45	-	61	=	77	M	93	]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o	127	[backspace]

Q&A
On Decision Making

- If
- Else if
- Else if
- …
- Else

Switch case statement

1. What is Switch statement?
2. Give an example using switch in C language

- Homework
- To be checked in Lab (next week)

Loop (iterator structure)

1. `for`
2. `while`
3. `do … while`

Iterator structure

Types of problems and solutions

- Suppose that we want to display a message “Hi, what is your name” and ask for names of 100 students
- Solution 1: Using 100 repeated instructions

```
var name: Sequence of character
begin
    write(“Hi, what is your name?”)
    read(name)
    write(“Hi, what is your name?”)
    read(name)
    ...
    write(“Hi, what is your name?”)
    read(name)
end
```

100 repeated instructions

- Solution 2 (Better solution): Use iteration structure (loop)
  - Loop allows a block of instruction/codes to be executed repeatedly within a condition

FOR ... DO

FOR loop

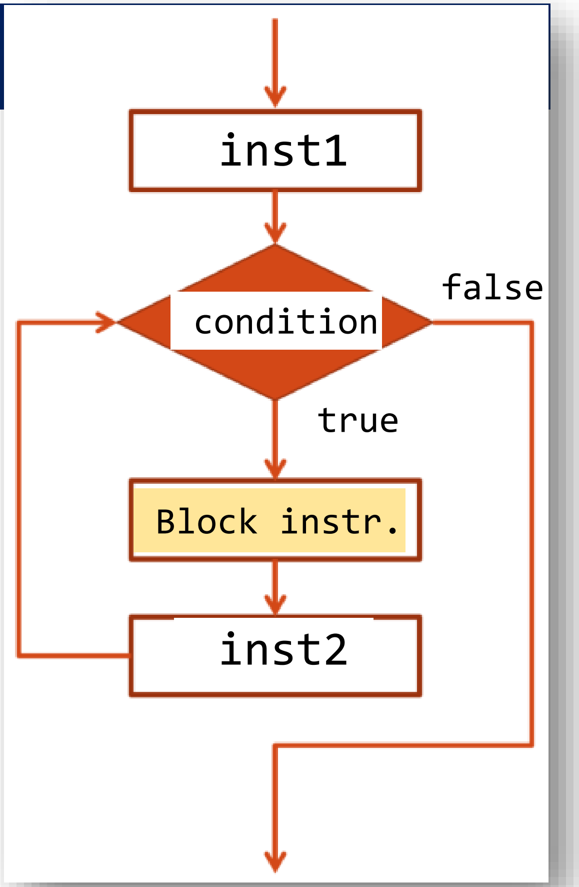
- Syntax:

instruction to initialize value of the control variable

loopback condition for stopping loop when it turns `false`

instruction to modify the value of control variable

```
for(inst1; condition; inst2) do
    block of instructions
end for
```



- Example:

```
var i: integer
begin
    for(i ← 25; i<=30; i ← i+1) do
        write(i, " ")
    end for
end
```

Output:

```
25 26 27 28 29 30
```

FOR ... DO

Examples

```
var i: integer
begin
    for(i ← 1; i<=7; i ← i+2) do
        write(i, “ ”)
    end for
end
```

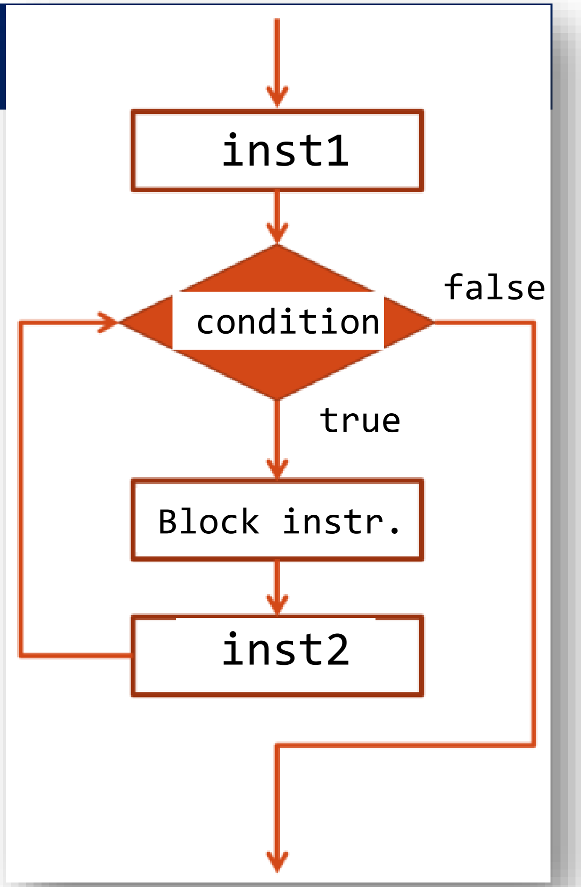
Output:

```
1 3 5 7
```

```
var i: integer
begin
    for(i ← 25; i>0; i ← i-5) do
        write(i, “ ”)
    end for
end
```

Output:

```
25 20 15 10 5
```



FOR ... DO

Examples

```
var i: integer
begin
    for(i ← 7 ; i>7; i ← i+2) do
        write(i, “ ”)
    end for
end
```

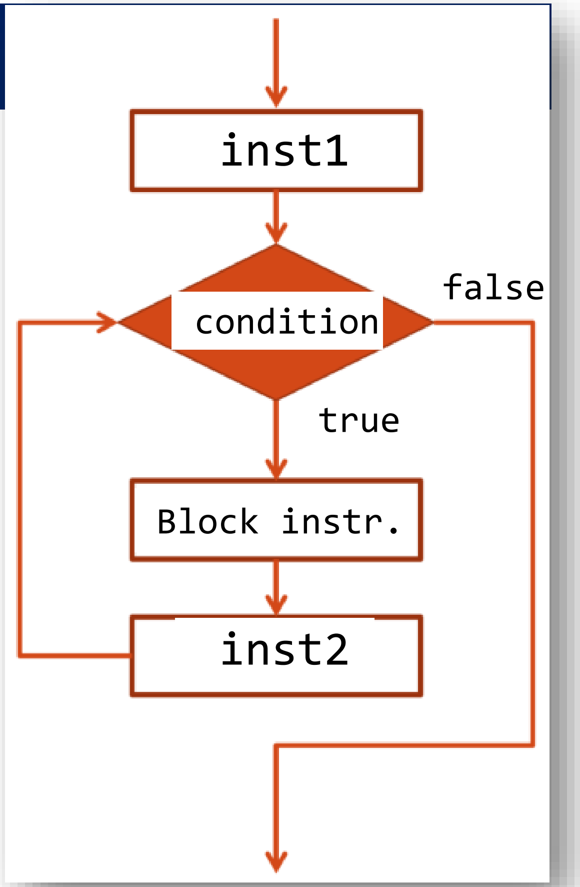
Output:

```
?
```

```
var i: integer
begin
    for(i ← 25; i>0; i ← i+1) do
        write(i, “ ”)
    end for
end
```

Output:

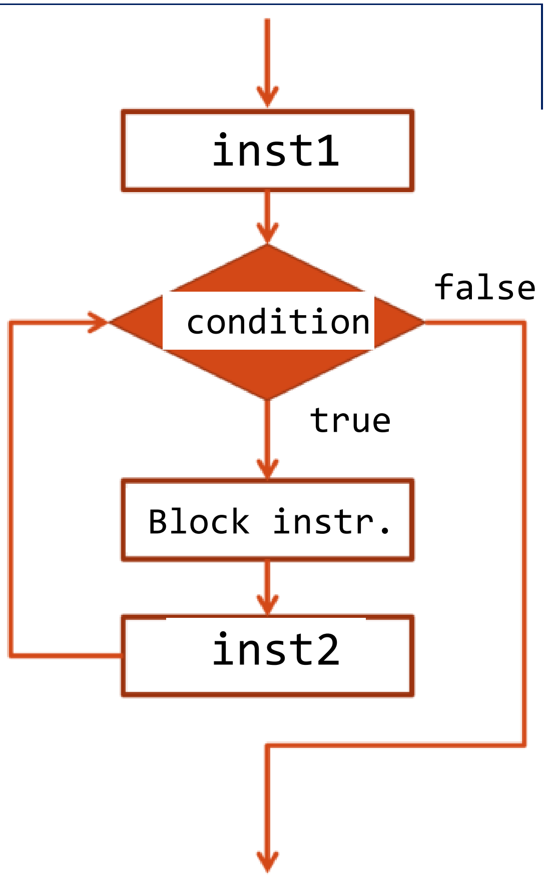
```
25 26 27 …
```



FOR ... DO

More examples

```
var i, j : integer
begin
    for(i ← 1; i <= 3; i ← i+1) do
        for(j ← 1; j <= 4; j ← j+1) do
            write(“A”)
            write(“B”)
        end for
        write(“ ”)
    end for
end
```



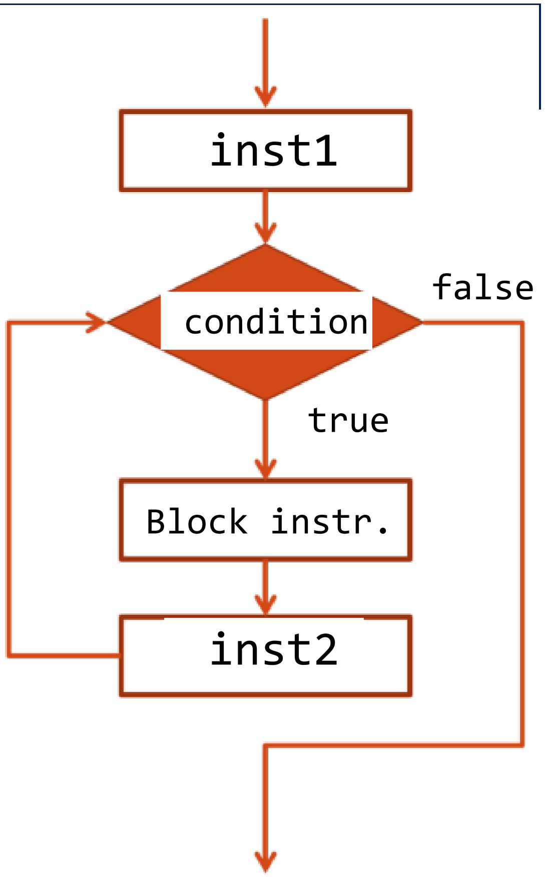
Output:

```
ABABABAB  ABABABAB  ABABABAB
```

FOR … DO

More examples

```
var i, j, k, n : integer
begin
    n ← 4
    for(i ← 1; i <= n; i ← i+1) do
        for(j ← 1; j <= n-i; j ← j+1) do
            write(“ ”)
        end for
        for(k ← 1; k <= 2*i-1; k ← k+1) do
            write(“*”)
        end for
        write(“\n”)
     end for
end
```



Output:

```
i=1     *
i=2    ***
i=3   *****
i=4  *******
```

Break 10mn
Start: 8:25am

Practice

Exercises on `switch … do`

1. Write an algorithm to show the result of students based on their grades. A user is required to input his/her grade.
   - Grade A : Very good
   - Grade B : Good
   - Grade C : Good enough
   - Grade D : Pass
2. Write an algorithm which allows a user to input 2 numbers and an operation (+, -, *, /) then using SWITCH … DO to perform the given operation between those two numbers.

Practice

Exercises on Loops

1. Write an algorithm to display the words “hello” 20 times and then “bye” 10 time. One line for displaying the word “hello”, and another line for displaying the word “bye”.
2. Write an algorithm to display all even numbers ***between*** 0 to 30.
3. Write an algorithm to calculate factorial of integer number n, where n is a positive number entered by a user.
4. Write an algorithm to sum suite number from 1 to n, where n is a positive number entered by a user.

Practice (loop)

Solve the below problem using loop:

- **For**
- **While**

Loop exercises: Write a C program to …

1. Display all numbers from 99 to 1.
2. Display all numbers from 1 to 100 except the number 50.
3. Display odd numbers between 8 to 1000 except the numbers 11, 17 and 21.
4. Show all integer divisible by 3 between 1 to 100 except 30, 60, and 90.
5. Sum all numbers from 1 to 100 then display the result.
6. Multiply all numbers from 1 to 100 then display the result.

TP

Loop exercises: Write a C program to …

7. Display the words “Hi” 20 times and then “bye” 10 time using **For loop**. One line for displaying the word “Hi”, and another line for displaying the word “bye”.
8. Display all even numbers ***between*** 0 to 30.
9. Calculate factorial of integer number n, where n is a positive number entered by a user.
10. Write an algorithm to sum suite number from 1 to n, where n is a positive number entered by a user.

Practices

Loop exercises: Write an algorithm to …

1. Compute and display the summation of the suit cube number starting from 1 up to n, where n is the input number entered by a user, n is greater than 1.

   Ex: Suppose the input is 3, then display 1^3 + 2^3 + 3^3 = 36
2. Check whether an input number is a primary number or not. The program runs indefinitely so that we can always check another input number.
3. Display all primary numbers in between 2 to 500.
4. Read 10 input numbers from a user and then find the maximum number and display it on screen.

TP

Loop exercises: Write a C program to …

1. Compute and display the summation of the suit cube number starting from n up to 1, where n is the input number entered by a user, n is greater than 1.

   Ex: Suppose the input is 3, then display $3^3 + 2^3 + 1^3 = 36$

2. Check whether an input number is a primary number or not. The program runs indefinitely so that we can always check another input number.

3. Display all primary numbers in between 2 to 500.

4. Read 20 input numbers from a user and then find the maximum number and display it on screen.

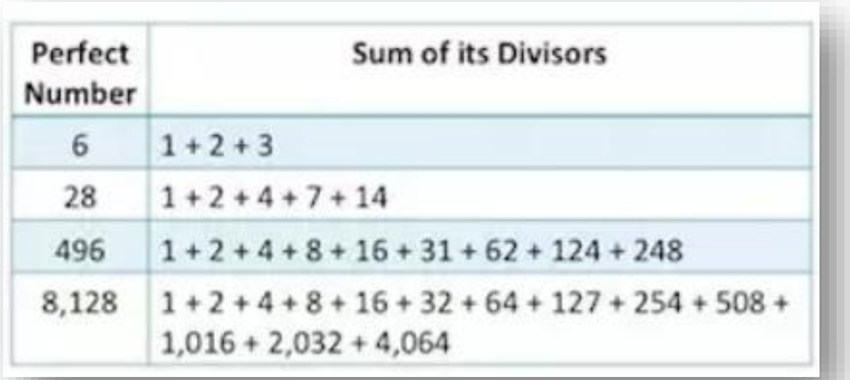
TP

Loop exercises: Write a C program to …

5. Check whether an input number is a **perfect number or not**. The program runs indefinitely so that we can always check another input number.

Perfect number is a positive integer that is equal to the sum of its proper divisors, excluding the number itself.
E.g: 6 is a perfect number.

- 6 has divisors 1, 2 and 3
- Since the sum of its divisors 1, 2, and 3 are equal to 6.



Perfect Number	Sum of its Divisors
6	1 + 2 + 3
28	1 + 2 + 4 + 7 + 14
496	1 + 2 + 4 + 8 + 16 + 31 + 62 + 124 + 248
8,128	1 + 2 + 4 + 8 + 16 + 32 + 64 + 127 + 254 + 508 + 1,016 + 2,032 + 4,064

TP

Loop exercises: Write a C program to …

6. Read 10 input numbers from a user and then find the minimum number and display it on screen.
7. Display the first n numbers of suit Fibonacci, where n is a number entered by a user.

The Fibonacci Sequence is the series of numbers: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...
The next number is found by adding up the two numbers before it.

- The 2 is found by adding the two numbers before it (1+1)
- The 3 is found by adding the two numbers before it (1+2),
- And the 5 is (2+3),
- and so on!

TP

Loop exercises: Write a C program to …

8. Ask a user to input many numbers as possible. When the user inputs 0, stop asking the user for the number and display the summation of all input numbers on the screen.

While loop

Activity

1) Team discussion on REVIEW LESSON

(7 students/team => Telegram, FB, …, Google Meet (G calendar))

Student list (Group A,B,C,D) Team1 (no. 1-7), Team2 (no. 8-15), ….

Remark: LESSONS REVIEW focus on What we have learnt so far (except Loop)

2) Quiz start at 10:30 - 11am (MS Team)

-individual

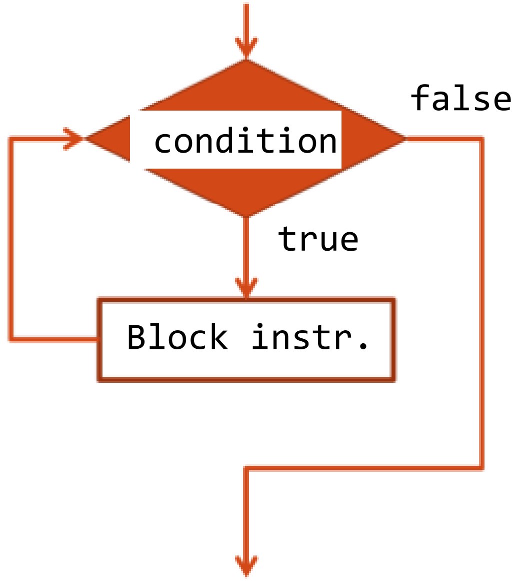
WHILE … DO

WHILE loop

- Syntax:

loopback condition for stopping loop when it turns `false`

```
while(condition) do
    //block instructions
end while
```



- **Block of instructions** is executed when condition is **true**
- Block of instructions is repeated to run until the condition is **false**
- <u>Note</u>:
    - Loop can be infinite loop if condition is wrong control
    - Block of instructions can control the `condition`

WHILE ... DO

WHILE loop

- Examples:

1

```
var n: integer
begin
    n ← 10
    while(n>0) do
        write(n, “ ”)
        n ← n-2
    end while
end
```

2

```
var n: integer
begin
    n ← 10
    while(n-1>2) do
        write(n+1, “ ”)
        n ← n-2
    end while
end
```

3

```
var n: integer
begin
    read(n)
    while(n>1) do
        write(n, “ ”)
        n ← n-2
    end while
end
```

Output:

```
10 8 6 4 2
```

Output:

```
11 9 7 5
```

Output:

```
?
```

WHILE ... DO

Examples

```
var n: integer
begin
    n ← 0
    while(n==0) do
        write(n, " ")
    end while
end
```

Output:

```
0
```

```
var n: integer
begin
    n ← 1
    while(n!=5) do
        write(n+1, " ")
        n ← n+1
    end while
end
```

Output:

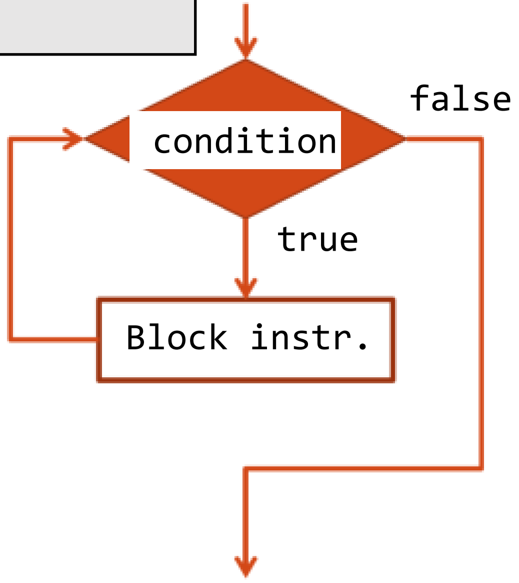
```
2 3 4 5
```

```
var n: integer
begin
        n ← 1
        while(n!=5) do
                write(++n, " ")
        end while
end
```

```
2 3 4 5
```

```
var n: integer
begin
        n ← 1
        while(n!=5) do
                write(n++, " ")
        end while
end
```

```
1 2 3 4
```



WHILE ... DO

Examples

```
var n: integer
begin
    read(n)
    while(n>1) do
        write(n, “ ”)
        n ← n+2
    end while
end
```

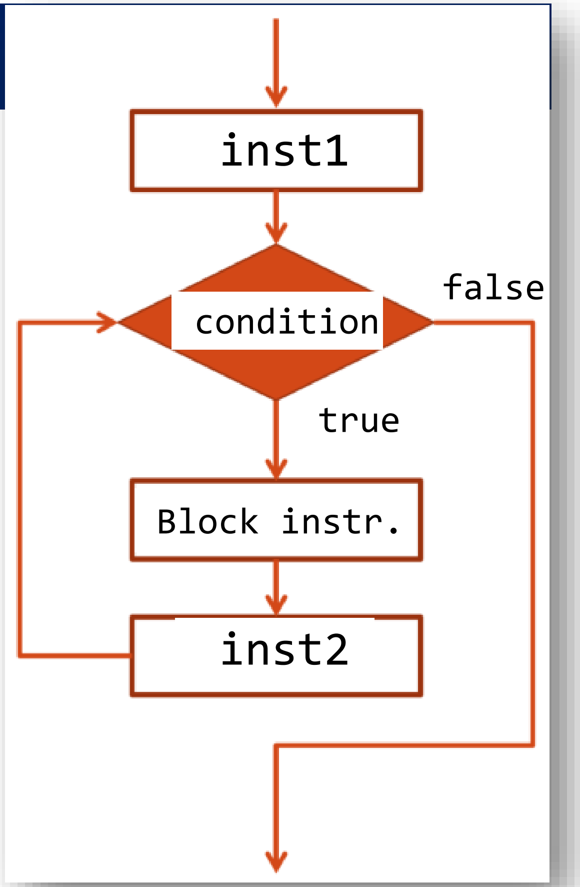
Output:

?

```
var n: integer
begin
    read(n)
    while(n>=1 AND n<50) do
        write(n, “ ”)
        n ← 2*n+1
    end while
end
```

Output:

?



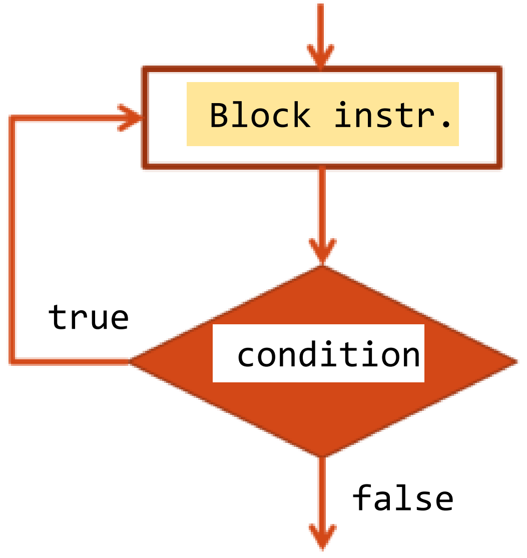
DO ... WHILE

DO WHILE loop

- Syntax:

loopback condition for stopping loop when it turns `false`

```
do
   block of instructions
while(condition)
```



- Instruction from block of instructions can control the `condition`
- The block of instructions can be executed at least once

DO ... WHILE

DO WHILE loop

- Examples:

1

```
var n: integer
begin
    n ← 10
    do
        write(n)
        n ← n-2
    while(n>0)
end
```

2

```
var n: integer
begin
    n ← -10
    do
        write(n)
        n ← n+2
    while(n<=0)
end
```

3

```
var n: integer
begin
    read(n)
    do
        write(n)
        n ← n-1
    while(n>0)
end
```

Output:

```
10 8 6 4 2
```

Output:

```
-10 -8 -6 -4 -2 0
```

Output:

```
?
```

Infinite loop

Notation on infinite loop

- Example:

```
var n : integer
begin
    n ← 10
    while(n>0) do
        write(n, “ ”)
    end while
end
```

```
var n : integer
begin
    n ← 10
    do
        write(n, “ ”)
        n ← n+3
    while(n>0)
end
```

Output:

```
10 10 …
```

Output:

```
10 13 16 …
```

Break Vs. Continue keyword

`break` statement breaks the loop/switch whereas

`continue` skip the execution of current iteration and continue to the next iteration (it does not break the loop/switch)

BREAK and CONTINUE statements

BREAK Vs. CONTINUE

- **Break**: allows to break/terminate the running loop
- **Continue**: allows to skip 1 iteration, so any instructions followed by **continue** will be skipped then the loop continue the next iteration
- Examples

```
var i : integer
begin
    for(i ← 1; i<=9; i ← i+1) do
        if (i==4) then
            continue
        end if
        write(i)
    end for
end
```

Output:

```
12356789
```

```
var i : integer
begin
    for(i ← 1; i<=10; i ← i+1) do
        if (i==3) then
            continue
        end if
        if (i==8) then
            break
        end if
        write(i)
    end for
end
```

Output:

```
124567
```

Syntax in C Programming

C program

Decision making: `if, else if, else`

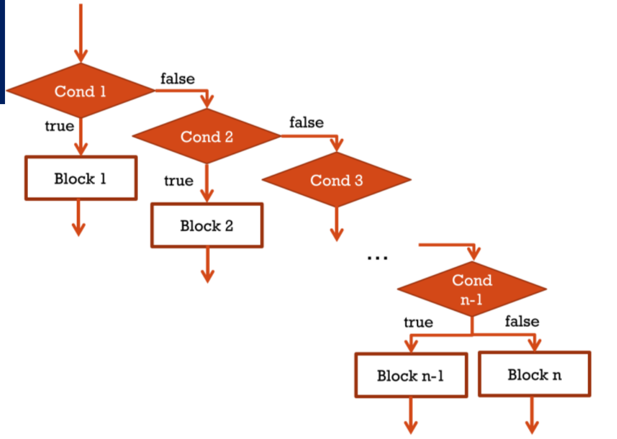
- Syntax

```
if (condition 1) {
     block of instructions 1
}else if (condition 2) {
     block of instructions 2
}else if (condition 3
     block of instructions 3
}
.
.
.
else if (condition n-1){
     block instruction n-1
}else{
     block of instruction n
}
```

Syntax in C

```
#include <stdio.h>
int main(){
     int n=10
     if (n==10) {
          print("n is equal to 10");
     }else if (n>0) {
          print("n is greater than 0");
     }else if (n<0) {
          print("n is less than 0");
     }else{
          print("n is 0");
     }
}
```

Example in C



```
if (condition 1) then
     block of instructions 1
else if (condition 2) then
     block of instructions 2
else if (condition 3) then
     block of instructions 3
.
.
.
else if (condition n-1) then
     block instruction n-1
else
     block of instruction n
end if
```

Algo syntax for decision making

C program

Decision making with choices: switch

- Syntax

```
switch (n){
    case constant1:
        // code to be executed if n is equal to constant1;
        break;

    case constant2:
        // code to be executed if n is equal to constant2;
        break;
        .
        .
        .
    default:
        // code to be executed if n doesn't match any constant
}
```

Syntax in C

```
#include <stdio.h>
int main(){
    char sex;
    printf("Enter your sex: ");
    scanf("%c", &sex);

    switch(sex){
     case 'M':
        printf("You are a male\n");
        break;
     case 'F':
        printf("You are a female\n");
        break;
     default:
        printf("wrong input\n");
    }
}
```

Example in C

C program

Loop: for

- Syntax

```
for(initStatement; condition; updateStatement){
      //codes
}
```

Syntax in C

1
```
#include <stdio.h>
int main(){
   int n,i;

   for(i=0; i<20; i++){
        printf("%d ", i);
   }
}
```

2
```
#include <stdio.h>
int main(){
   int n;

   for(int i=0; i<20; i++){
        printf("%d ", i);
   }
}
```

Examples of using **for** loop in C

3
```
#include <stdio.h>
int main(){
   int num;
   printf("Enter a number: ");
   scanf("%d", &num);

   for(int i=0; i<num; i++){
        printf("%d ", i);
   }
}
```

Output:

```
0 1 2 … (num-1)
```

4
```
#include <stdio.h>
int main(){
   int num;
   printf("Enter a number: ");
   scanf("%d", &num);

   for(int i=num; i>0; i--){
        printf("%d ", i);
   }
}
```

Output:

```
num (num-1) … 1
```

5
```
#include <stdio.h>
int main(){
   int num;
   printf("Enter a number: ");
   scanf("%d", &num);

   for(int i=0; i<num; i--){
        printf("%d ", i);
   }
}
```

Output (infinite loop)**:**

```
0 -1 -2 …
```

TD: Practices on Control structure (if, else if, else, switch, loops)

Write an algorithm for each of the following problem below:

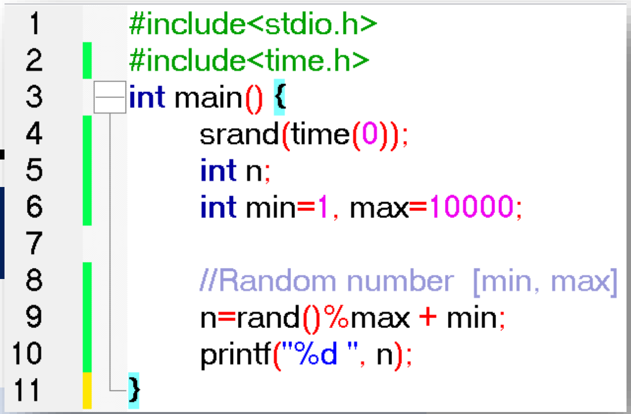
1. Write an algorithm to show the result of students based on their grades. A user is required to input his/her grade.
   - Grade A is “Excellent”, Grade B is “Very good”, Grade C is “Good”, Grade D is “Fair”, Grade F is “Fail”, other grades display “Invalid grade”
2. Write an algorithm which allows a user to input 2 numbers and an operation (+, -, *, /) then using SWITCH … DO to perform the given operation between those two numbers.
3. Display all numbers from 99 to 1.
4. Display all numbers from 1 to 100 except the number 50.
5. Display odd numbers between 8 to 1000 except the numbers 11, 17 and 21
6. Show all integer divisible by 3 between 1 to 100 except 30, 60, and 90.
7. Read 10 input numbers from a user and then find the minimum. Then display the minimum number.
8. Read 20 input numbers from a user and then find the maximum number. Then display the maximum number.
9. Test if a given integer number is a primary number. **Hint**: A primary number is a number that is divisible only by itself.
10. Write an algorithm to get an input of a day in a number (1 to 7) then display the day in word (ex: 1=Monday, 2: Tuesday, …etc.). When user input other numbers (not in 1 to 7), display “Invalid”
11. Write an algorithm to display the cube of number starting from 1 up to a given integer
    - Ex: input is 5, then display $1^3 + 2^3 + 3^3 + 4^3 + 5^3$
    - Or this format is okay: 1^3+2^3+3^3+4^3+5^3
12. Write an algorithm to display a string in a reverse order (Ex: welcome => emoclew)
13. Write an algorithm to add all the numbers input by a user until the user inputs zero. Notice that a number is input one by one until user inputs zero, then display the summation and stop the program.

Challenge!

Period: 40mn

Kahoot!
10:40am

Tip: To generate a random number

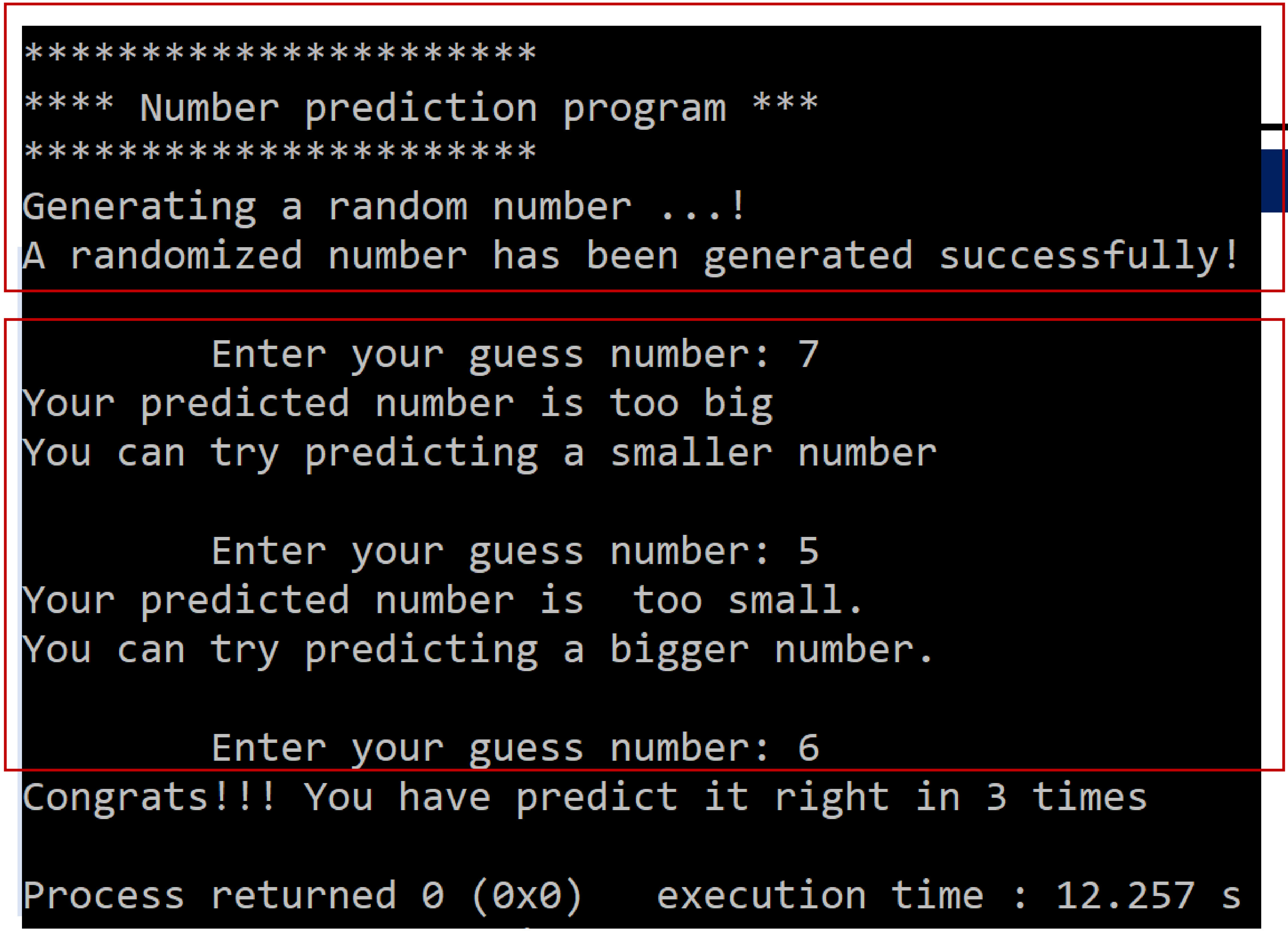


```
#include<stdio.h>
#include<time.h>
int main() {
        srand(time(0));
        int n;
        int min=1, max=10000;

        //Random number  [min, max]
        n=rand()%max + min;
        printf("%d ", n);
}
```

Number prediction program!

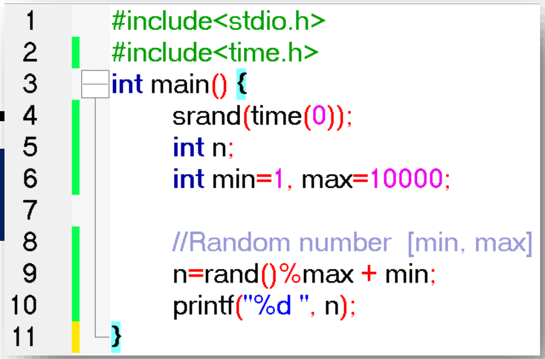
Write a C program to guess a number. The computer generate a random number. Then program asks a user to input a number for guessing. The program keeps asking the user to input a number until the user input the correct one compared to the randomized number.

- If the user inputs a number **greater than the randomized number**, tell a user to input another smaller number.
- If the user inputs a number **less than the randomized number**, tell a user to input another bigger number.
- If the user inputs **the correct number (the number is same to the randomized number)**, display "Congratulations! You guess only **n** times to be correct.", where n is the number of attempts the user made to get it right.



```
**********************
**** Number prediction program ***
**********************
Generating a random number ...!
A randomized number has been generated successfully!

        Enter your guess number: 7
Your predicted number is too big
You can try predicting a smaller number

        Enter your guess number: 5
Your predicted number is  too small.
You can try predicting a bigger number.

        Enter your guess number: 6
Congrats!!! You have predict it right in 3 times

Process returned 0 (0x0)   execution time : 12.257 s
```

Tips to generate a random number



```
#include<stdio.h>
#include<time.h>
int main() {
      srand(time(0));
      int n;
      int min=1, max=10000;

      //Random number  [min, max]
      n=rand()%max + min;
      printf("%d ", n);
}
```

Quiz

kahoot.it