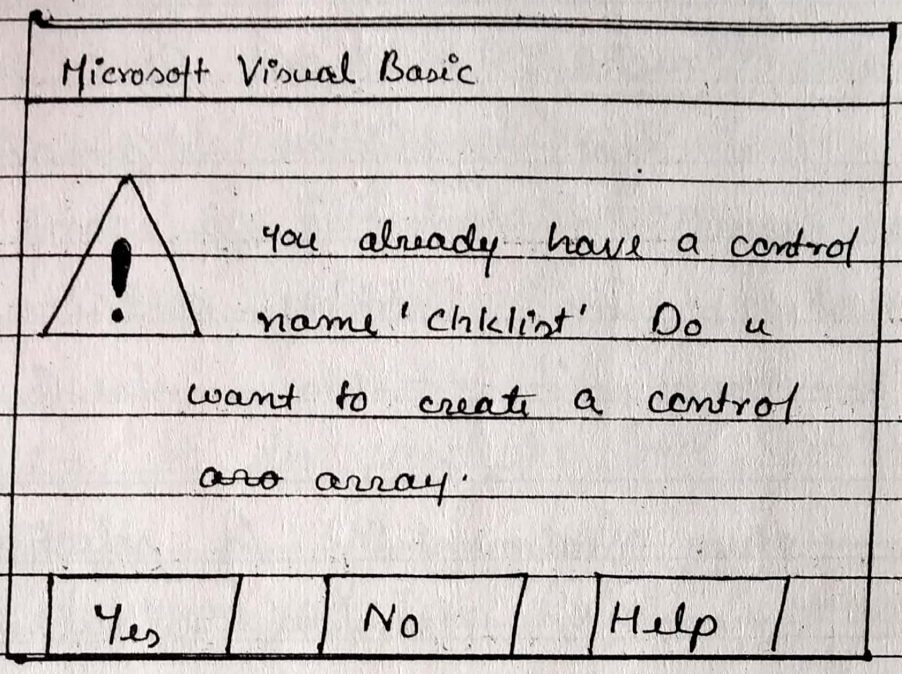
'chklist' at the second check box control specify the same name as the first that is 'chklist', we will get a Message as given below.




Click on Yes Button to create a control array.

## UNIT-4

11/8/12

### * Control Structure :-

Control Structure allows us to control the flow of our program structure. If we don't use control structure the program logic will flow through statements from left to right & top to bottom. Most of the power & utility of any programming language lies within decision structure & looping. The control

Structure used in V.B are
1) Decision Structure 2) Loop Structure.

1) Decision Structure :- Visual Basic procedure can test condition depending upon the result of test condition if perform diff. operation. The Decision Structure include if--- then ; if --- then --- else & select--case

i) if ----then Statement :- A selection structure or decision structure is used to select among alternative courses of action.
The if--- then selection structure either perform an action if the condition is True or skip the action if the condition is False. The if--- then structure is called Single selection structure because it select or ignore a single structure because it select or ignore a single action. In V.B we use 2 types of if--- then structure.

1) Single Line Structure.
2) Multi-Line Structure.

1) Single-Line Structure :-

In Single-Line structure the action is given on the same line of condition checking. Here, no need to give an if statement to terminate if--- then statement.

Syntax :- IF (test condition Then <statement>)

Example :-

if (num > 0) Then Print "Number is Positive".

2) Multi-Line Structure :-

In Multi-Line Structure a block of statement is attached with if --- then statement. We must give "End if" statement at the last.

Syntax :-

IF (test condition) Then

<Block of statement>

end if

Example:-

```
IF (num > 0) Then
Printf ("Number is +ve")
endif.
```

13/8/12

ii) If --- then --- else ---)

A variation of if --- then --- else is it perform an action. If a condition is true & performs a diff. action if the condition is False. The if --- then --- structure is also called "Double Selection structure" because it performs two diff. actions. The if --- then structure performs an indicated action only when the condition is True otherwise the action is skipped whereas if --- then --- else selection structure allow the programmer to specify that a diff. action is to be performed when the condition is true & then the condition is false.

Syntax :-

```
if (condition) then
    ≡≡≡
Statement block 1
    ≡≡≡
else
    ≡≡≡
Standard block 2
    ≡≡≡
end if
```

Example :-

```
if (marks > 60)
Print "passed"                          ┐ Statement
Print "Allow to promote in next sem"     ┘ block
else
Print "fail"  ] Statement block-2
endif
```

```
Private sub command1-click()
Dim marks as integer
marks = val (Text1.text)
if (marks>60)
Print "passed"
Print "allow to promote in next sem."
else
Print "fail"
endif.
```

Another variation of if - then - else selection structure uses "elseif" keyword.

Syntax:-

```
if (condition) Then
    Statement block-1
elseif (condition 2)
    statement-block 2
elseif (condition-n)
    statement-block n
else
    default-Block
endif
```

Example:-

```
if (marks >= 75) then
    Print "Excellent"
elseif (marks >= 60) then
    Print "first"
elseif (marks >= 40) then
    Print "Pass"
else
Print "fail"
endif.
```

* IIF-Statement :- It stands for immediate if statement. It is closely related to if-- then --- else statement. IIF statement works on 3 expressions:-

1) The 1st expression represents the condition.
2) The 2nd " is process when condition is True.
3) The 3rd expression is process when condition is False.

Note:- The 1st expression is evaluated if condition is True then the value of expression 2 will be the value of whole expressions otherwise, the value of expression 3 will be the value of whole expression.

Syntax:- IIF (<exp 1>, <exp 2>, <exp 3>)

Example:-

Str = IIF ((marks)=60), "Pass", "Fail")

we can also use nesting of IIF statement by placing another IIF statement at 2nd expression or 3rd expression.

## Select-case Statement :-

V.B provides select case structure as an alternative to if-- then --- else structure. The select case statement an execute one block of statement from multiple-block of statement. The select case is similar to if-- then --- else statement but it make code more readable or reliable then if-- then --- else.

A select-case structure works with single test expression that evaluate once at the top of the structure. V.B then compares the result of this expression with the value for each cases in the select-case structure. If there is a match it execute the block of statement associated with that case

### Syntax:-

Select case <Test expression>
[case Experiment-1 <statement-1>]
[case Experiment-2 <statement-2>]

[case Experiment-n <statement-n>]

[case else <else statement>]
End SELECT

Example:-

```
Marks = Marks/10
Select case Marks
Case Is 7   print "Excellent"
Case 6:  Print "First"
Case 4 to 5  Print "Second".
Case is 3  print "pass"
case else  print "Fail"
end select.
```

* Looping :-) Loop Structure allowed us to execute one or more lines of code repeatedly. V.B supports the following loop structure.

1) Do---loop :-) The Do--- loop execute a block of statement as long as condition is true.
It works in two ways as specified syntax.

i) Entry Controlled Loop :-) In Entry controlled loop 1st we check the

condition if we get it true then execute statement block within the loop. It close with loop keyword.

Syntax:-

Do while <condition>
<statement_block>
Loop

Example:-

```
x = 1
Dim x as integer
Do while (x <= 10)
Print x
x = x + 1
loop.
```

iv) ~~Exit controlled loop~~

Until Keyword

Do-loop also works with until keyword but until works as well as condition is False. When condition becomes true, it terminates the loop

Syntax :-

```
                    F
do until <condition>
<statement - Block>
loop   T
```

Example :-

```
x=1
do until (x=10)
Print x
x= x+1
loop.
```

ii) Exit Controlled Loop :-

It execute the statement block 1st & then check the condition at the exit time of Loop so, it is known as exit-controlled loop.

Syntax :-

```
Do
<statement - Block>
while (Test - condition)
```

Example :-

```
x=1
Do
Print x
x= x+1
while (x<=10)
end sub.
```

Teacher Signature _____

Page: / /

• Until keyword :-

Syntax :-

```
do
<statement - Block>
until (Test condition)
```

Exaupli :-

```
x=1
do
Print x
x=x+1
until (x=11)
```

2) While --- wend :-

The while --- wend execute a block of statement while the condition is True. If the condition is True all the statement in blocks are executed & when then the wend keyword reached the control resumes to the while statement which evaluates condition again.

If the condition is False the wend program execute the statement following the wend statement.

Syntax :-

WHILE (Test-condition)

Statement - Block

wend

Example :-

```
i=1
while (i<=10)
Print i
i= i+1
wend
```

16/8/12

## For ---- Next :-

The most basic type of Loop in V.B is the For--- Next loop. We use it to execute the statement for specified number of times. A For --- Next loop uses a variable called counter that increases or decreases in value during each repetation of loop.

Syntax :-

For counter = start to End [Step Increment / Decrement

Statement Block

Next [counter variable]

Example :-

```
For i=0 to 10
  Print i
Next.
```

Note :- The increment argument can be positive or negative. If increment is positive start must be less than or equals to end. If increment is (-ve) (decrement) start must be greater or equals to end. If step increment is not set (The increment is by default to 1.

No

Note :- In the syntax argument counter, start, end & step increment represents a number.

## UNIT-5

* Procedure :-

The application is made up of small self-contained segment. These small segments are called procedures. There are two types of Procedures in V.B

→ Sub-routine / Procedure :-

A sub-routine contains a block of code or a block of statement that carried out a well-defined