Decision making and branching

CHAPTER 26

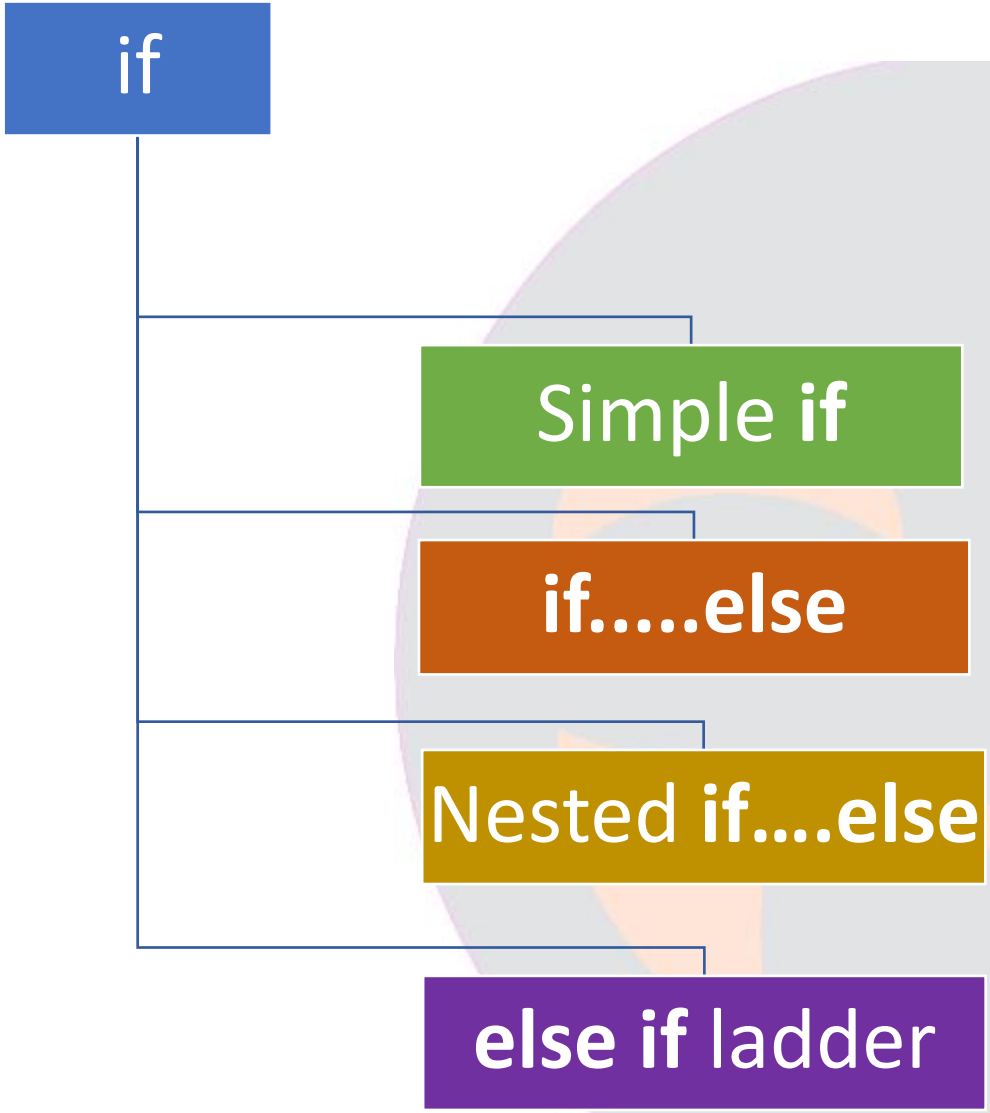
SURESH TECHS

C PROGRAMMING COURSE

Decision making statements

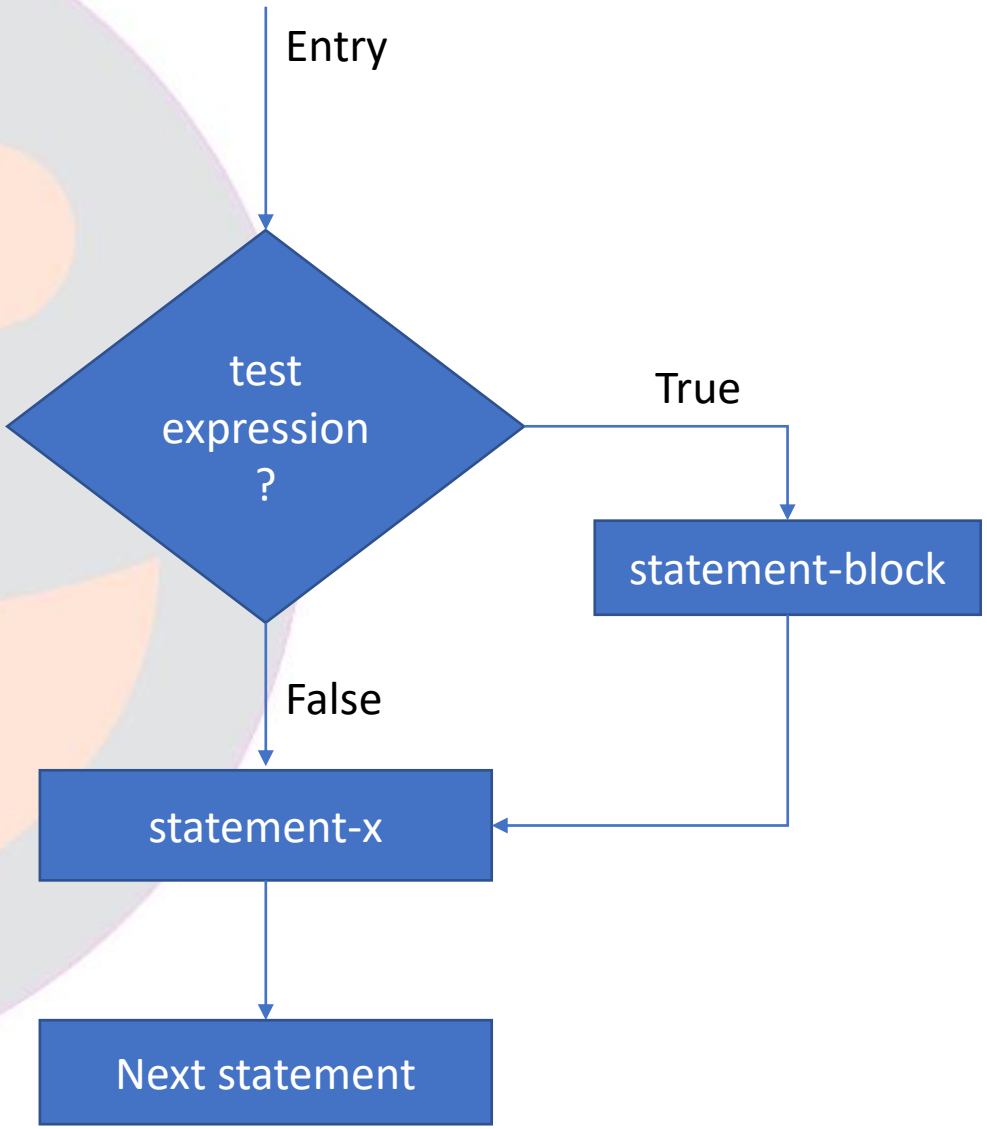
```
if ( accident ) {
 Take turn
}
```

1. if
2. switch
3. conditional operator
4. goto


if
Simple if
if.....else
Nested if....else
else if ladder

Simple if

```
if ( test-expression ){
   statement-block
}
statement-x;
```



Simple **if**

```c
#include<stdio.h>
int main(){
    int rank;
    printf("Welcome suresh\n");
    printf("What is your eamcet rank?");
    scanf("%d",&rank);
    printf("Your rank is : %d",rank);
    return 0;
}
```

```
Welcome suresh
What is your eamcet rank?49000
Your rank is : 49000
```

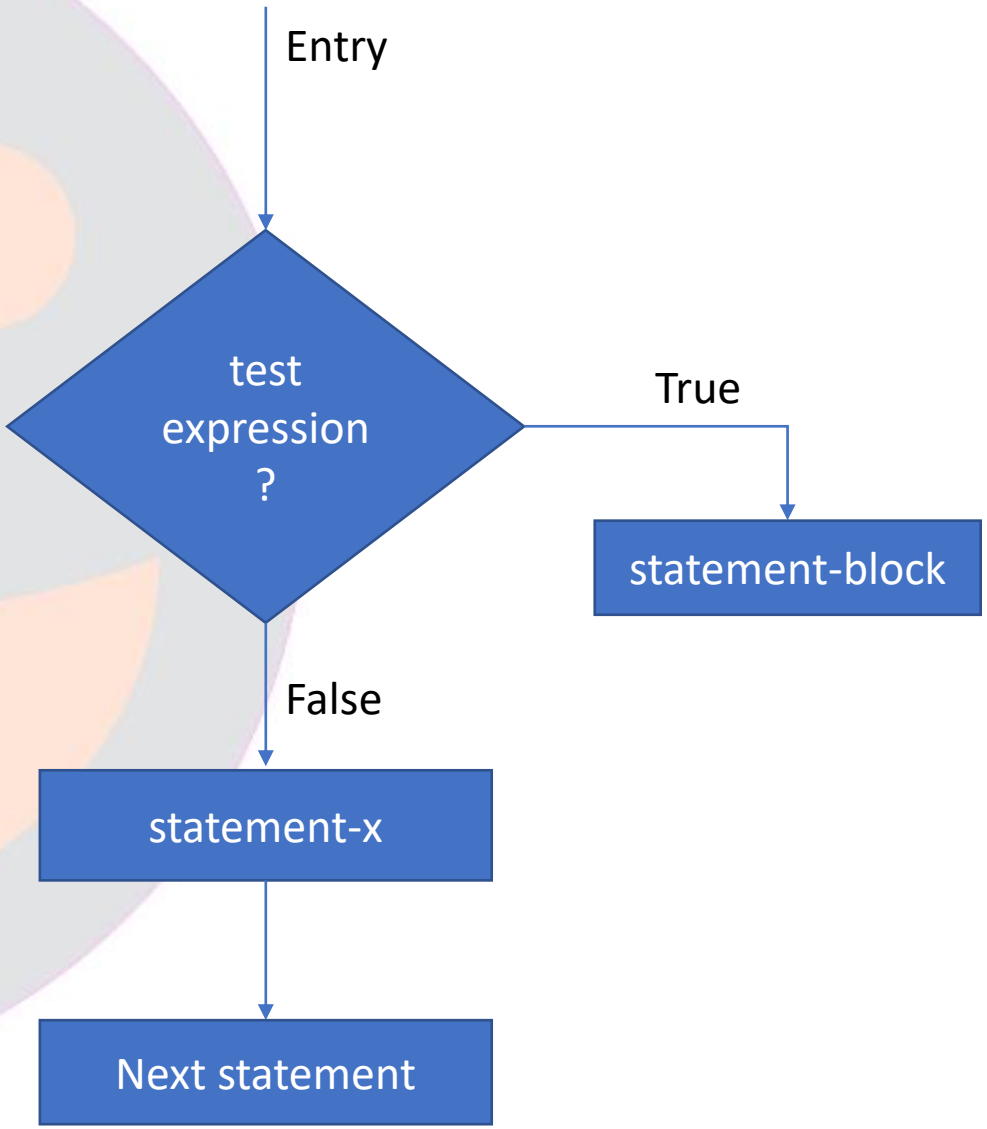
Write a program to print “Very Good” **if** the rank is less than 30000

```c
#include<stdio.h>
int main(){
    int rank;
    printf("Welcome suresh\n");
    printf("What is your eamcet rank?");
    scanf("%d",&rank);
    printf("Your rank is : %d",rank);
    return 0;
}
```

```
Welcome suresh
What is your eamcet rank?49000
Your rank is : 49000
```

Simple **if**

```c
#include<stdio.h>
int main(){
    int rank;
    printf("Welcome suresh\n");
    printf("What is your eamcet rank?");
    scanf("%d",&rank);
    if(rank<30000){
        printf("Very Good ");
    }
    printf("Your rank is : %d",rank);
    return 0;
}
```



We can have any number of if statements

```c
#include<stdio.h>
int main(){
    int rank;
    printf("Welcome suresh\n");
    printf("What is your eamcet rank?");
    scanf("%d",&rank);
    if(rank<30000){
        printf("Very Good\n");
    }
    if(rank<5000){
        printf("Super ra\n");
    }
    if(rank<1000){
        printf("Excellent ra\n");
    }
    if(rank<50){
        printf("Nuvvu turumu ra");
    }
    printf("Your rank is : %d",rank);
    return 0;
}
```

```
Welcome suresh
What is your eamcet rank?49000
Your rank is : 49000
```

```
Welcome suresh
What is your eamcet rank?3500
Very Good
Super ra
Your rank is : 3500
```

All of the if statements are checked

No need to put brackets if you have **single statement**

```c
#include<stdio.h>
int main(){
    int rank;
    printf("Welcome suresh\n");
    printf("What is your eamcet rank?");
    scanf("%d",&rank);
    if(rank<30000)
        printf("Very Good\n");

    if(rank<5000)
        printf("Super ra\n");

    if(rank<1000){
        printf("Excellent ra\n");
    }
    if(rank<50)
        printf("Nuvvu turumu ra");
        printf("Nuvve turumu ra..");

    printf("Your rank is : %d",rank);
    return 0;
}
```

"Nuvve turumu ra" would print because it is out of the if condition

We can have **if inside if**

```c
#include<stdio.h>
int main(){
    int rank;
    printf("Welcome suresh\n");
    printf("What is your eamcet rank?");
    scanf("%d",&rank);
    if(rank<30000){
        printf("Very Good\n");
        if(rank<5000){
            printf("Super ra\n");
        }
        if(rank<1000){
            printf("Excellent ra\n");
            if(rank<50){
                printf("Nuvvu turumu ra");
            }
        }
    }
    printf("Your rank is : %d",rank);
    return 0;
}
```

```
Welcome suresh
What is your eamcet rank?500
Very Good
Super ra
Excellent ra
Your rank is : 500
```

Nested if

Difference between these?

```
Welcome suresh
What is your eamcet rank?50000
Your rank is : 50000
```

```c
#include<stdio.h>
int main(){
    int rank;
    printf("Welcome suresh\n");
    printf("What is your eamcet rank?");
    scanf("%d",&rank);
    if(rank<30000){
        printf("Very Good\n");
    }
    if(rank<5000){
        printf("Super ra\n");
    }
    if(rank<1000){
        printf("Excellent ra\n");
    }
    if(rank<50){
        printf("Nuvvu turumu ra");
    }
    printf("Your rank is : %d",rank);
    return 0;
}
```

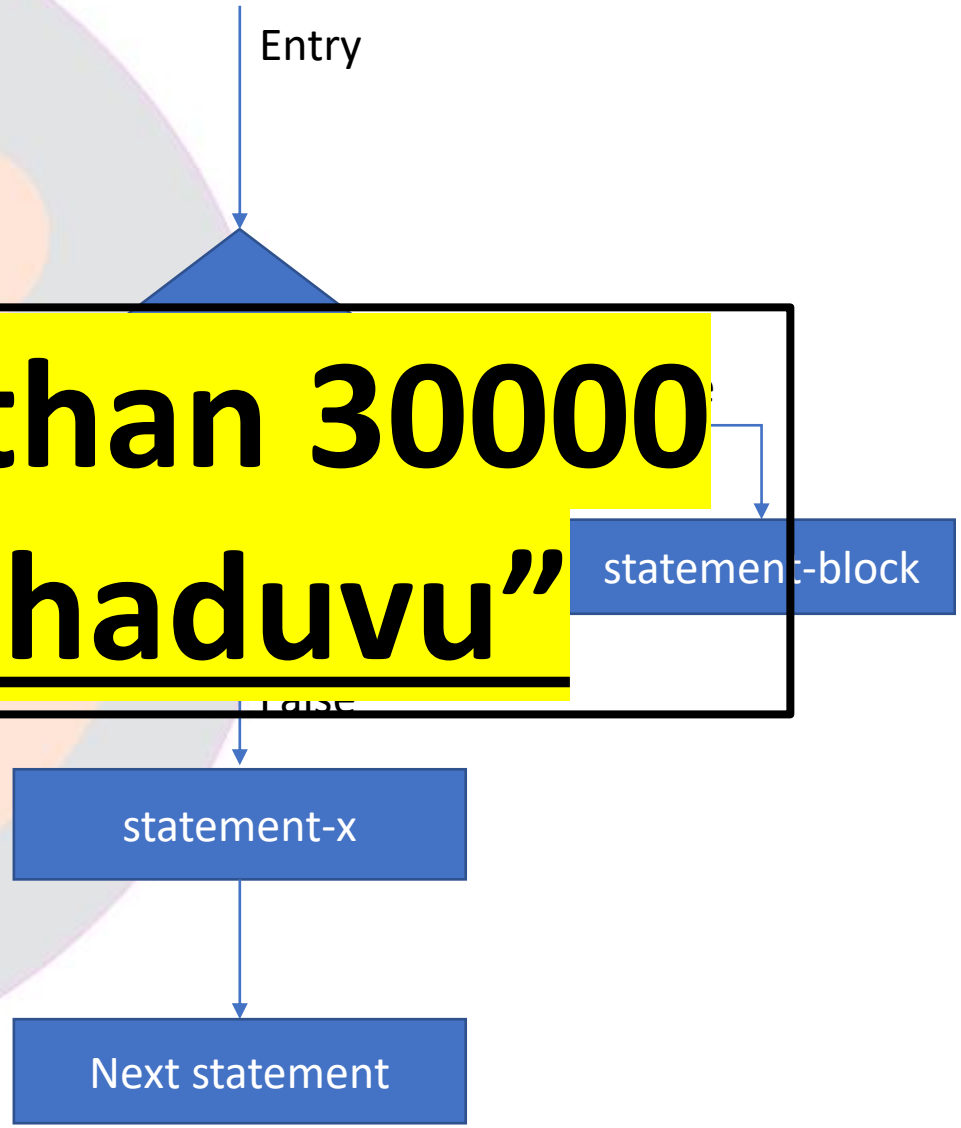
```c
#include<stdio.h>
int main(){
    int rank;
    printf("Welcome suresh\n");
    printf("What is your eamcet rank?");
    scanf("%d",&rank);
    if(rank<30000){
        printf("Very Good\n");
        if(rank<5000){
            printf("Super ra\n");
        }
        if(rank<1000){
            printf("Excellent ra\n");
            if(rank<50){
                printf("Nuvvu turumu ra");
            }
        }
    }
    printf("Your rank is : %d",rank);
    return 0;
}
```

Simple if

```
#include<stdio.h>
int main(){
    if(ra
        p
    }
    printf("Your rank is : %d",rank);
    return 0;
}
```

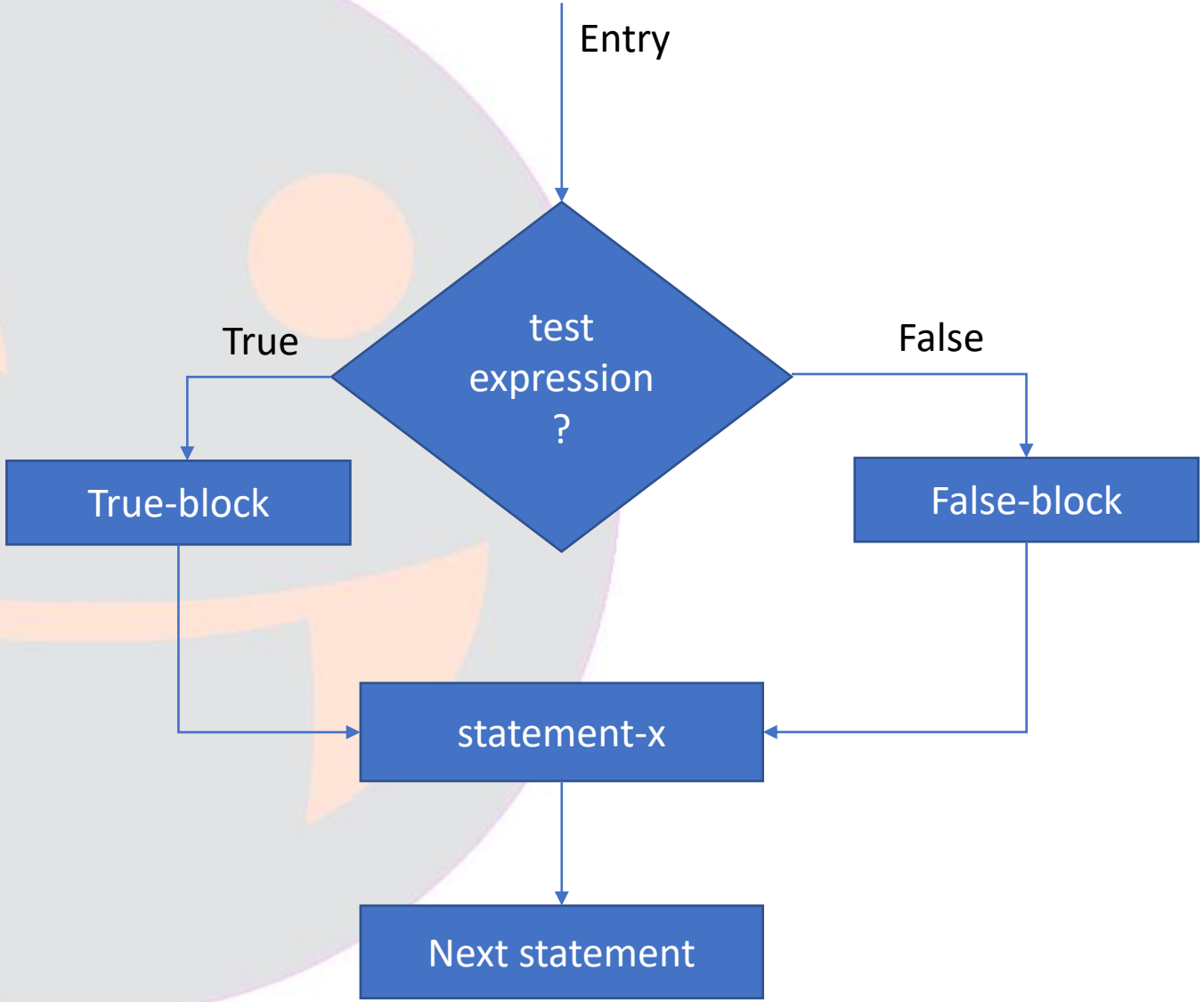
if the rank is more than 30000

“kunchum baga chaduvu”



if else

```
if ( test-expression ){
   true-block statement(s)
}else{
  false-block statement(s)
}
statement-x;
```



Either True block or False block but not both

if....else

```
#include<stdio.h>
int main(){
    int rank;
    printf("Welcome suresh\n");
    printf("What is your eamcet rank?");
    scanf("%d",&rank);
    if(rank<30000){
        printf("Very Good ");
    }else{
        printf("Kunchum baga chaduvu ");
    }
    printf("Your rank is : %d",rank);
    return 0;
}
```

Short form for **if else**

```
if (test-expression){
  true-block statement
}else{
  false-block statement
}
```

Conditional operator

Since they always start with a condition as the first operand

Ternary operator

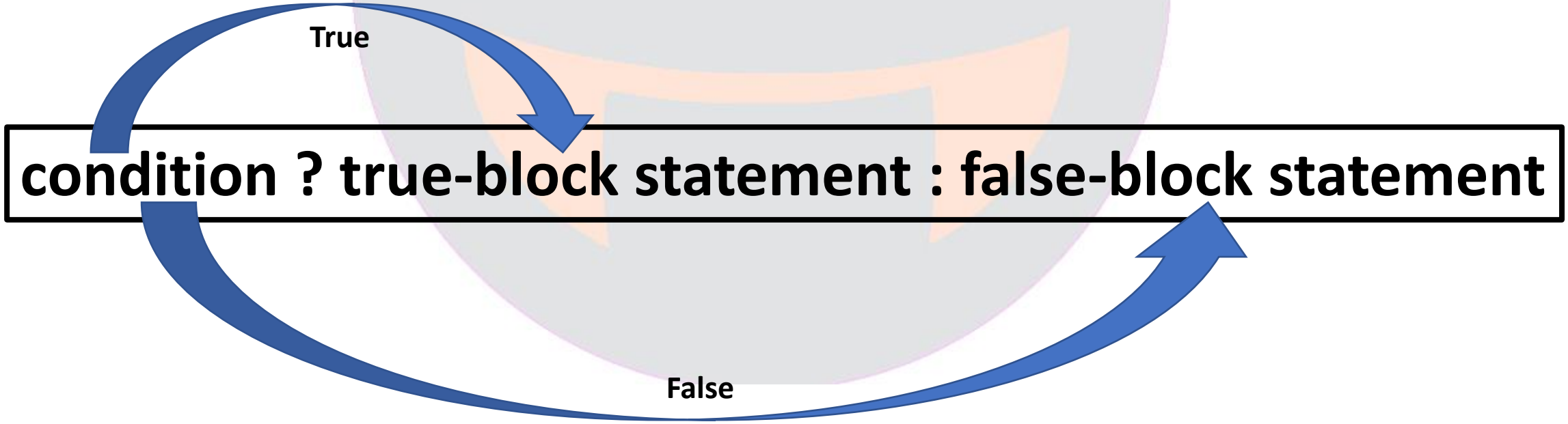
Since there are three operands

test-expression ? true-block statement : false-block statement

Works only if there is a single statement for if and else blocks

Conditional/ternary operator

- The conditional operator takes an expression and **executes the first statement if the expression evaluates to be true**, and the **second statement if the expression evaluates to be false**
- Any non-zero value will be considered as true, and 0 as false



Conditional operator/ternary operator

```c
#include<stdio.h>
int main(){
    int num;
    printf("Enter a number: ");
    scanf("%d",&num);
    if(num%2==0){
        printf("Even number");
    }else{
        printf("Odd number");
    }
    return 0;
}
```

```c
#include<stdio.h>
int main(){
    int num;
    printf("Enter a number: ");
    scanf("%d",&num);
    /*if(num%2==0){
        printf("Even number");
    }else{
        printf("Odd number");
    }*/
    num%2==0? printf("Even number"): printf("Odd number");
    return 0;
}
```

Test

- Write a program to print even or odd number
- Check if the number divides by 5 when it is an even number

- Ex: 20
  - Even number and divides by 5
- Ex: 34
  - Even number and not divisible by 5
- Ex: 15
  - Odd number

Check if it divides by 5 in case it is an even number

```c
#include<stdio.h>
int main(){
    int num;
    printf("Enter a number: ");
    scanf("%d",&num);
    if(num%2==0){
        printf("Even number\n");
        if(num%5==0){
            printf("and divides by 5");
        }else{
            printf("and not divides by 5");
        }
    }else{
        printf("Odd number");
    }
    return 0;
}
```

How to implement it using conditional operator 🤔?

We can nest conditional operators 😍 😍 😍

```c
#include<stdio.h>
int main(){
    int num;
    printf("Enter a number: ");
    scanf("%d",&num);
    if(num%2==0){
        printf("Even number\n");
        if(num%5==0){
            printf("and divides by 5");
        }else{
            printf("and not divides by 5");
        }
    }else{
        printf("Odd number");
    }
    return 0;
}
```

```c
#include<stdio.h>
int main(){
    int num;
    printf("Enter a number: ");
    scanf("%d",&num);
    num%2==0? num%5==0? printf("Even number\n and
divides by 5"):printf("Even number\n and not divides by
5") : printf("Odd number");

    return 0;
}
```

Another use/version of **conditional operator**

- **variable = condition ? value1: value2**
- Shorter version of **itself**

```c
#include<stdio.h>
int main(){
    int num;
    int money;
    printf("Enter a number: ");
    scanf("%d",&num);
    money = num>50? num+100 : num+50;
    printf("money is: %d",money);
    return 0;
}
```

```c
#include<stdio.h>
int main(){
    int num;
    int money;
    printf("Enter a number: ");
    scanf("%d",&num);//user entered 55
    money = num>50? printf("Welcome") : num+50;
    printf("money is: %d",money);
    return 0;
}
```

Difference between if else and conditional operator

Conditional operator in C	if-else statement in C
The conditional operator is a **single programming statement** and **can only perform one operation**.	The if-else statement is a **block statement**, you can group **multiple statements using a parenthesis**.
The conditional operator can return a value and so can be **used for performing assignment operations**.	The if else statement does not return any value and **cannot be used for assignment purposes**.
The **nested** ternary operator is **complex and hard to debug**.	The **nested** if-else statement is **easy to read and maintain**.

What have we completed so far?

```c
#include<stdio.h>
int main(){
    int rank;
    printf("Welcome suresh\n");
    printf("What is your eamcet rank?");
    scanf("%d",&rank);
    if(rank<30000){
        printf("Very Good ");
    }
    printf("Your rank is : %d",rank);
    return 0;
}
```

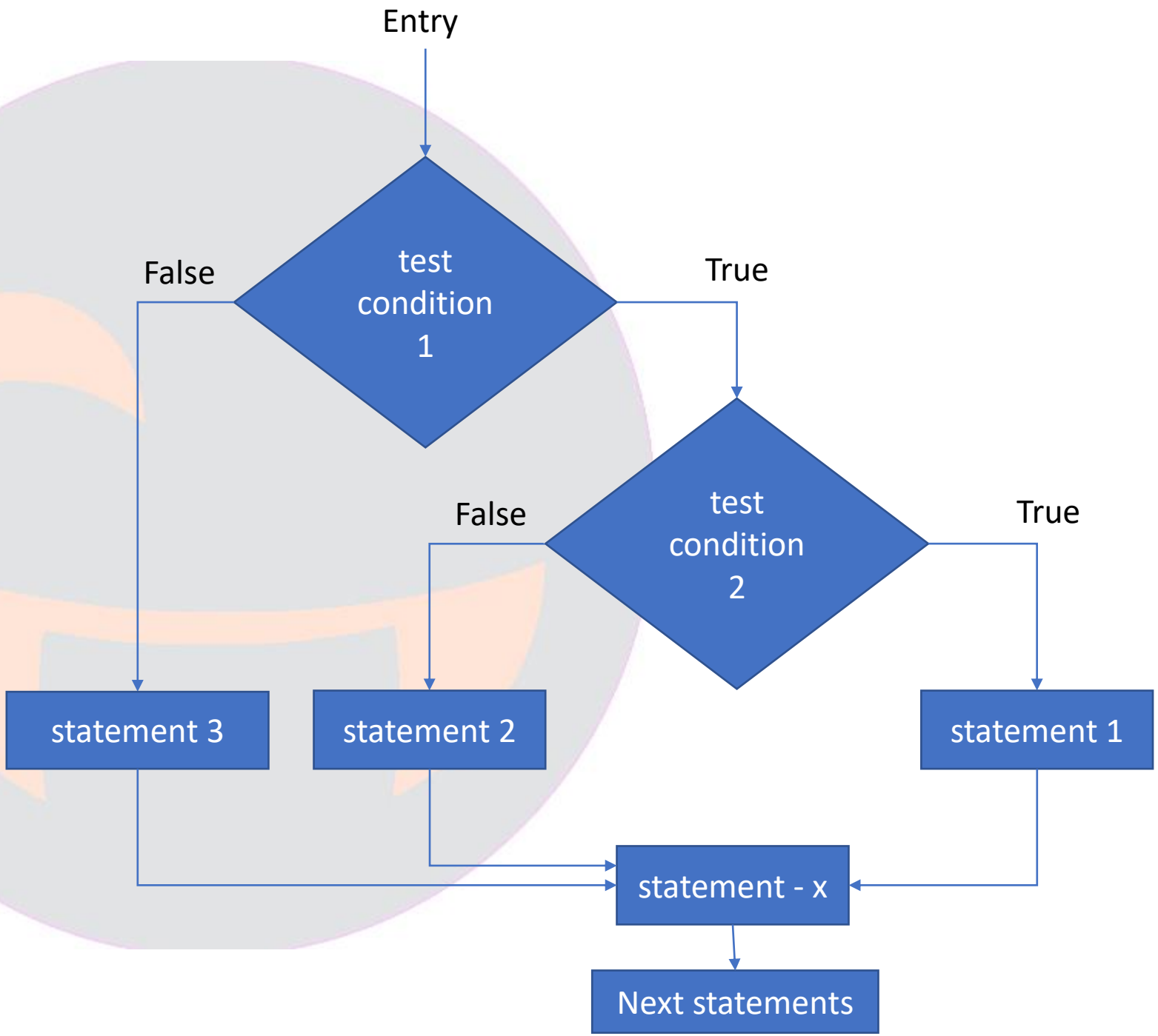
```c
#include<stdio.h>
int main(){
    int num;
    printf("Enter a number: ");
    scanf("%d",&num);
    /*if(num%2==0){
        printf("Even number");
    }else{
        printf("Odd number");
    }*/
    num%2==0? printf("Even number"): printf("Odd number");
    return 0;
}
```

```c
#include<stdio.h>
int main(){
    int rank;
    printf("Welcome suresh\n");
    printf("What is your eamcet rank?");
    scanf("%d",&rank);
    if(rank<30000){
        printf("Very Good ");
    }else{
        printf("Kunchum baga chaduvu ");
    }
    printf("Your rank is : %d",rank);
    return 0;
}
```

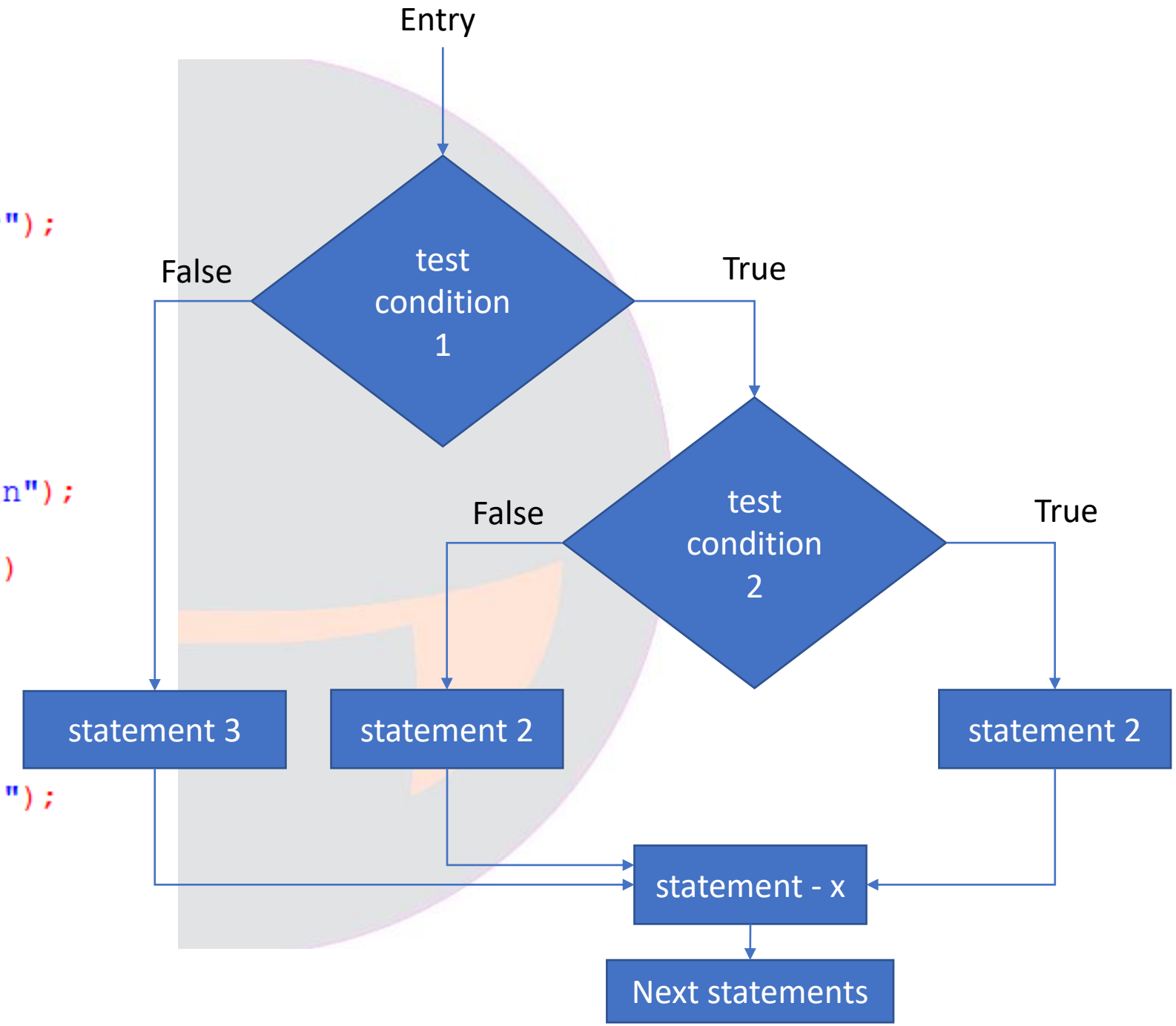
```c
#include<stdio.h>
int main(){
    int rank;
    printf("Welcome suresh\n");
    printf("What is your eamcet rank?");
    scanf("%d",&rank);
    if(rank<30000){
        printf("Very Good\n");
        if(rank<5000){
            printf("Super ra\n");
        }
        if(rank<1000){
            printf("Excellent ra\n");
            if(rank<50){
                printf("Nuvvu turumu ra");
            }
        }
    }
    printf("Your rank is : %d",rank);
    return 0;
}
```

Nested **if else**

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



```c
#include<stdio.h>
int main(){
    int rank;
    printf("Welcome suresh\n");
    printf("What is your eamcet rank?");
    scanf("%d",&rank);
    if(rank<30000){
        printf("Very Good \n");
        if(rank<5000){
            printf("Super ra \n")
            if(rank<1000){
                printf("brilliant ra\n");
            }else{
                printf("excellent ra")
            }
        }else{
            printf("Nice ra\n")
        }
    }else{
        printf("Kunchum baga chaduvu ");
    }
    printf("Your rank is : %d",rank);
    return 0;
}
```



else if ladder

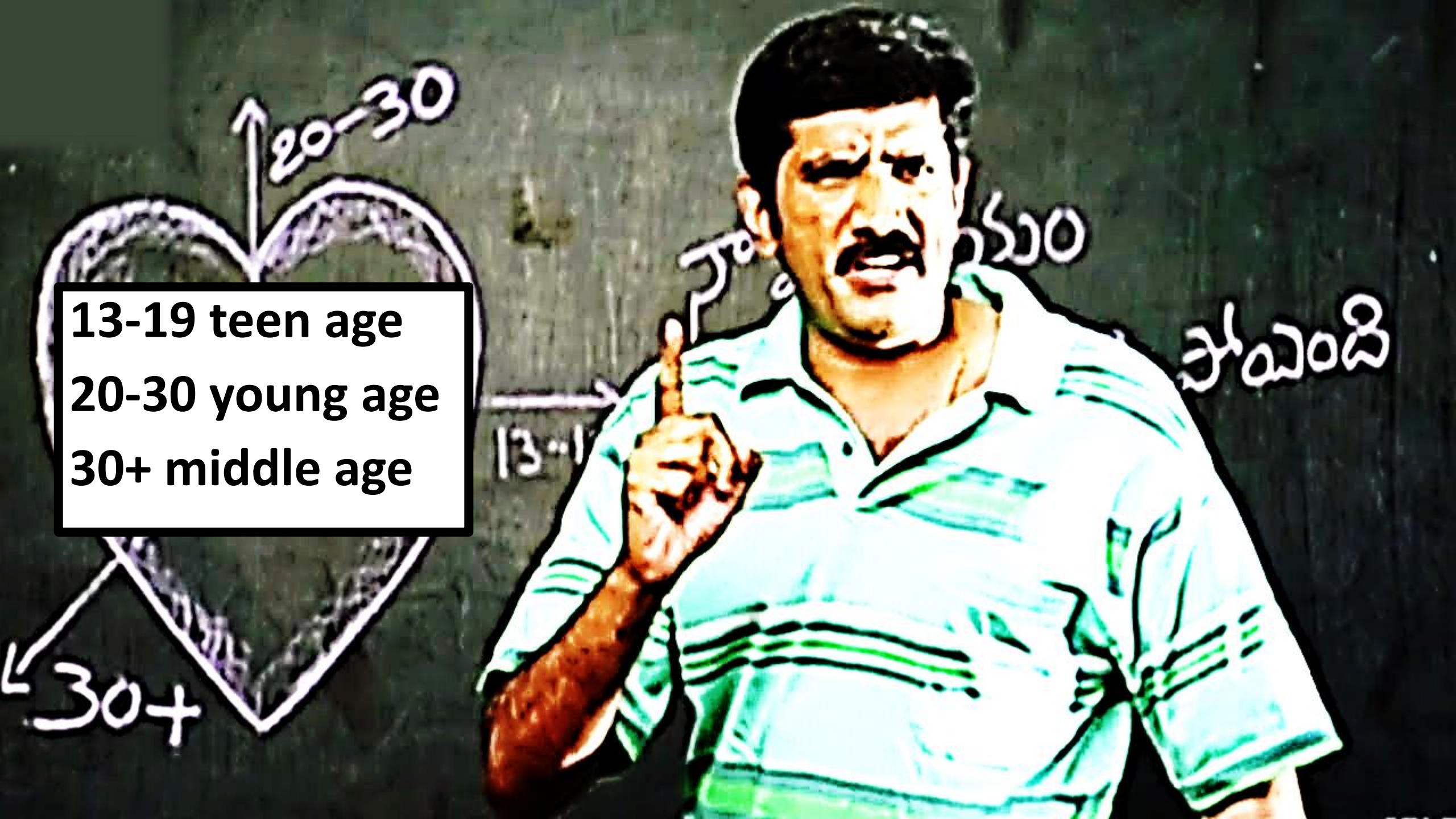
```c
#include<stdio.h>
int main(){
    int rank;
    printf("Welcome suresh\n");
    printf("What is your eamcet rank?");
    scanf("%d",&rank);
    if(rank<30000){
        printf("Very Good\n");
    }
    if(rank<5000){
        printf("Super ra\n");
    }
    if(rank<1000){
        printf("Excellent ra\n");
    }
    if(rank<50){
        printf("Nuvvu turumu ra");
    }
    printf("Your rank is : %d",rank);
    return 0;
}
```

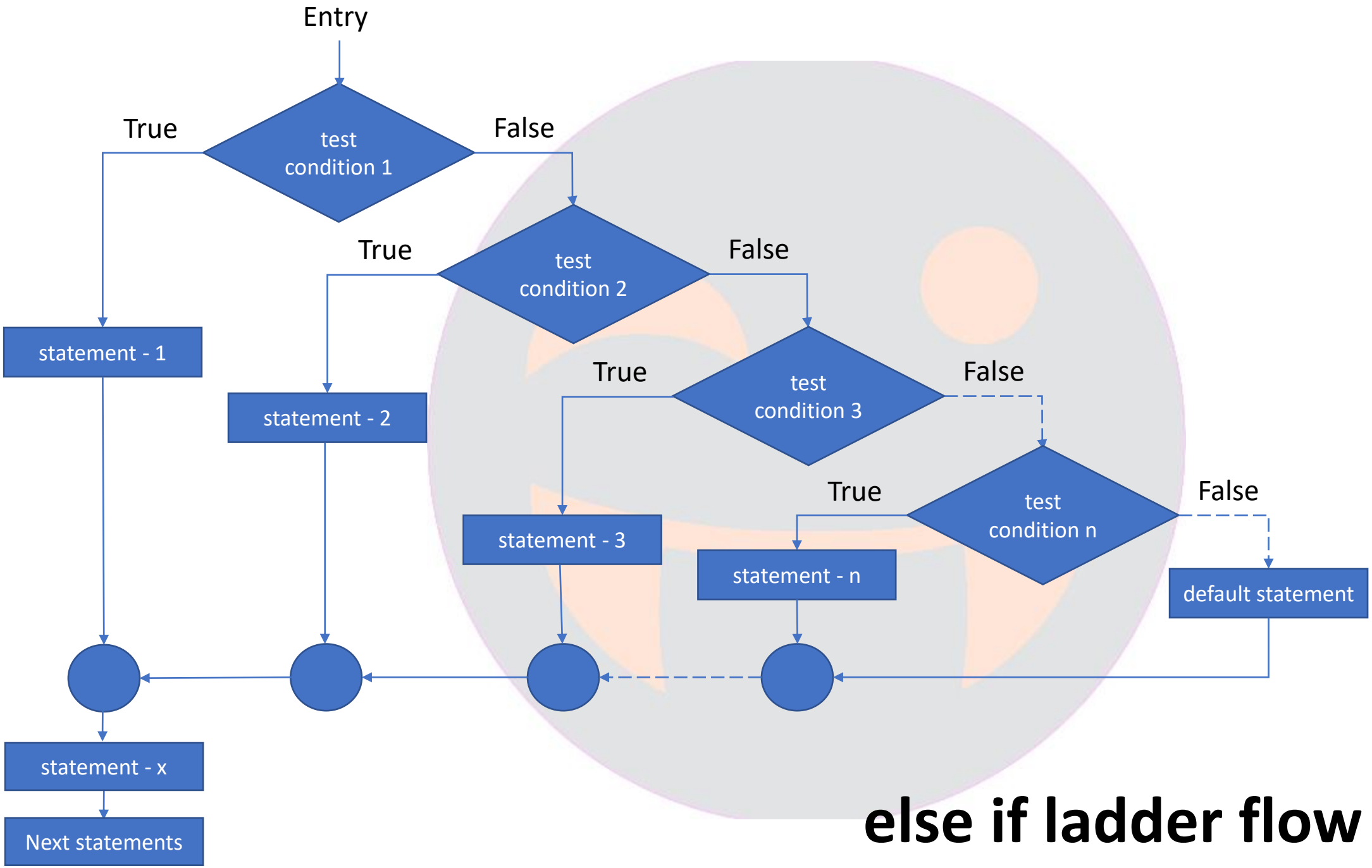
```c
#include<stdio.h>
int main(){
    int rank;
    printf("Welcome suresh\n");
    printf("What is your eamcet rank?");
    scanf("%d",&rank);
    if(rank<50){
        printf("Nuvvu turumu ra\n");
    }
    else if(rank<1000){
        printf("Excellent ra\n");
    }
    else if(rank<5000){
        printf("Super ra\n");
    }
    else if(rank<30000){
        printf("Good ra");
    } else{
        printf("Baga chaduvukooo");
    }
    printf("Your rank is : %d",rank);
    return 0;
}
```

Only one set of statements gets executed at any point of time

```
if ( test-condition 1 )
   statement-1;
else if( test-condition 2)
    statement-2
else if( test-condition-3)
    statement-3
else if( test-condition-4)
    statement-4
else
  default-statement
statement-x;
```

```c
#include<stdio.h>
int main(){
    int rank;
    printf("Welcome suresh\n");
    printf("What is your eamcet rank?");
    scanf("%d",&rank);
    if(rank<50){
        printf("Nuvvu turumu ra\n");
    }
    else if(rank<1000){
        printf("Excellent ra\n");
    }
    else if(rank<5000){
        printf("Super ra\n");
    }
    else if(rank<30000){
        printf("Good ra");
    } else{
        printf("Baga chaduvukooo");
    }
    printf("Your rank is : %d",rank);
    return 0;
}
```


20-30
13-19 teen age
20-30 young age
30+ middle age
30+



else if ladder flow chart

Switch

- Program looks complex when there are **many number of else if condition**
- To reduce the complexity of **else if** ladder, **switch** statement was introduced

case value must be an integer or character constant

Expression should either be integer expression or character

```
if ( test-condition 1 )
   statement-1;
else if( test-condition 2)
    statement-2
else if( test-condition-3)
    statement-3
else if( test-condition-4)
    statement-4
else
  default-statement
statement-x;
```

```
switch (expression) {
  case value-1:
        statements
  case value-2:
        statements
  case value-3:
        statements
  ........
  ........
  default:
        default statements
}

statement-x
```

Switch

- The switch statement tests the **value of a given variable**(or **<u>expression</u>**) against a list of **case values** and **when a match is found**, a **block of statements** associated with that **case** is **executed**.

```
switch (expression) {
   case value-1:
          statements
   case value-2:
          statements
   case value-3:
          statements
   ........
   ........
   default:
          default statements
}

statement-x
```

But the output is different right?

```c
#include<stdio.h>
int main(){
    int studyClass;
    printf("Which class admission do you want? ");
    scanf("%d",&studyClass);
    if(studyClass==6){
        printf("Go to first floor first room");
    }
    else if(studyClass==7){
        printf("Go to first floor second room");
    }
    else if(studyClass==8){
        printf("Go to second floor first room");
    }
    else if(studyClass==9){
        printf("Go to second floor second room");
    }
    else if(studyClass==10){
        printf("Go to third floor");
    }else{
        printf("Sorry, we offer 6th to 10th class");
    }
    return 0;
}
```

```c
#include<stdio.h>
int main(){
    int studyClass;
    printf("Which class admission do you want? ");
    scanf("%d",&studyClass);
    switch(studyClass){
        case 6:
            printf("Go to first floor first room");
        case 7:
            printf("Go to first floor second room");
        case 8:
            printf("Go to second floor first room");
        case 9:
            printf("Go to second floor second room");
        case 10:
            printf("Go to third floor");
        default:
            printf("Sorry, we offer 6th to 10th class");

    }
    return 0;
}
```

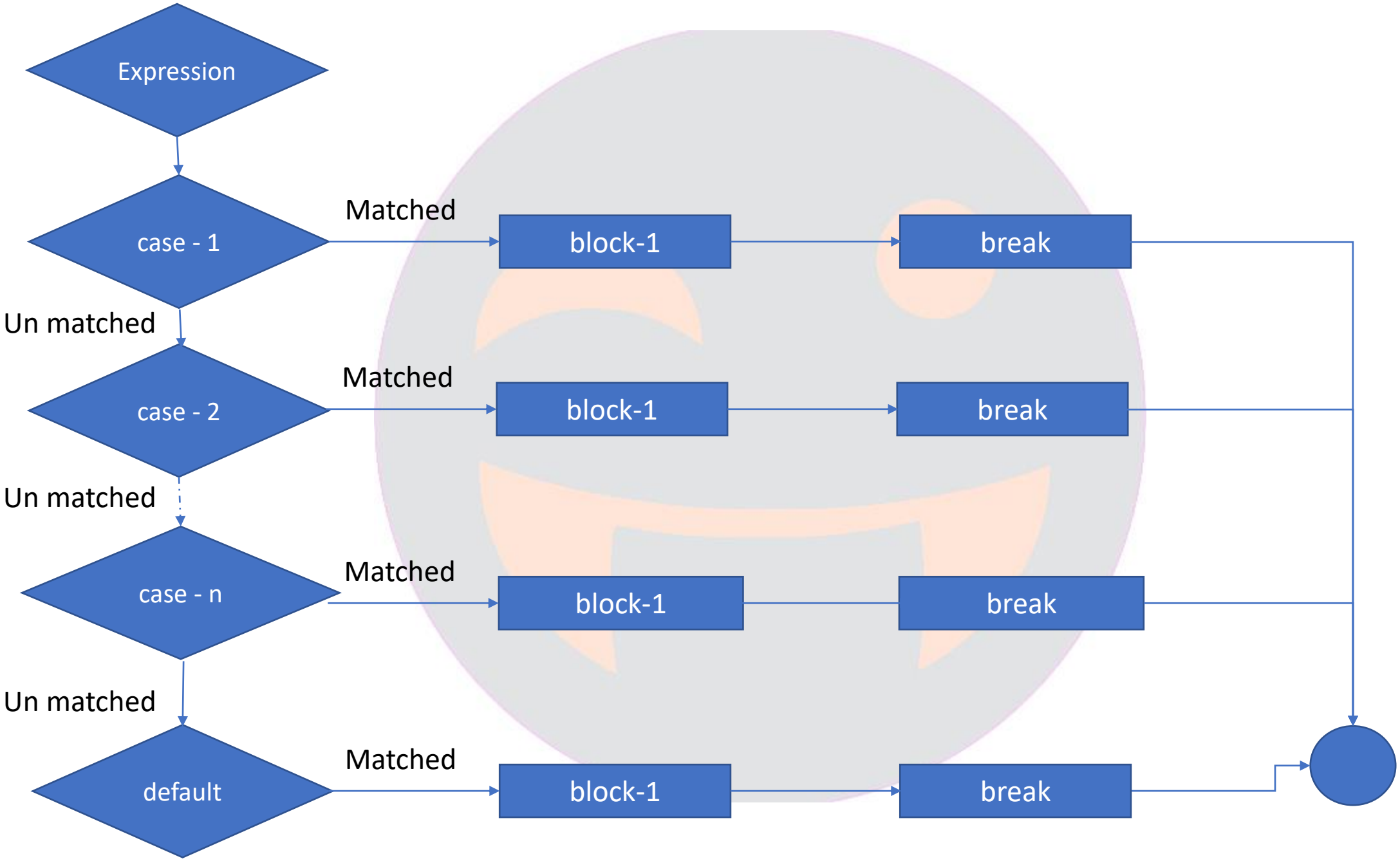
We need to stop(break) the execution

```c
#include<stdio.h>
int main(){
    int studyClass;
    printf("Which class admission do you want? ");
    scanf("%d",&studyClass);
    switch(studyClass){
        case 6:
            printf("Go to first floor first room");
            break;
        case 7:
            printf("Go to first floor second room");
            break;
        case 8:
            printf("Go to second floor first room");
            break;
        case 9:
            printf("Go to second floor second room");
            break;
        case 10:
            printf("Go to third floor");
            break;
        default:
            printf("Sorry, we offer 6th to 10th class");
            break;
    }
    return 0;
}
```

1. The **break statement** in switch case is **not must**. **It is optional**.

2. If there is **no break statement** found in the case, **all the cases will be executed** present after the matched case.

3. It is known as **<u>fall through state</u> of C switch statement**


Expression
case - 1
Matched
block-1
break
Un matched
case - 2
Matched
block-1
break
Un matched
case - n
Matched
block-1
break
Un matched
default
Matched
block-1
break

Tell me the first character in your name, I will tell you what you are

```
What is the first letter in your name? s
People whose name starts with "S" are multi-talented and can shine in any field whether it is acting, politics, business
, sports or any creative field. These people have only one motive in life i.e to achieve success, fame, money. Thus, the
se people put in a lot of hard work to gain success professionally. However, when it comes to love, you will often see t
hem standing alone or ditched by someone
```

Let’s write the program

I will provide the notes, don’t worry

```
switch (expression) {
   case value-1:
          statements
           break;
   case value-2:
          statements
            break;
   case value-3:
          statements
            break;
   ........
   ........
   default:
          default statements
            break;
}
statement-x
```

First, the **expression** inside the switch clause is evaluated to an integral constant.

Its result is then compared against the **case-value inside each case statement**

If a match is found, all the statements following that matching case label are executed, **until a break** or end of switch is encountered. **This is a critical statement**.

Small test for you

```c
#include<stdio.h>
int main(){
    int num=5;
    switch(num){
    case 1:
        printf("one\n");
        break;
    case 3:
        printf("three\n");
        break;
    case 5:
        printf("five\n");
    case 2:
        printf("love\n");
    case 6:
        printf("zero");
    }
    return 0;
}
```

```
five
love
zero
```

```c
#include<stdio.h>
int main(){
    int num=5;
    switch(num){
    case 1:
        printf("one\n");
        break;
    case 3:
        printf("three\n");
        break;
    case 5:
        printf("five\n");
    case 2:
        printf("love\n");
    case 6:
        printf("zero\n");
    default:
        printf("welcome");
    }
    return 0;
}
```

```
five
love
zero
welcome
```

```c
#include<stdio.h>
int main(){
    int num=10;
    switch(num){
    case 1:
        printf("one\n");
        break;
    case 3:
        printf("three\n");
        break;
    case 5:
        printf("five\n");
    case 2:
        printf("love\n");
    case 6:
        printf("zero\n");
    default:
        printf("welcome");
    }
    return 0;
}
```

```
welcome
```

```c
#include<stdio.h>
int main(){
    int num=1;
    switch(num){
    case 1:
        printf("one\n");
        break;
    case 3:
        printf("three\n");
        break;
    case 5:
        printf("five\n");
    case 2:
        printf("love\n");
    case 6:
        printf("zero\n");
    default:
        printf("welcome");
    }
    return 0;
}
```

```
one
```

What if I want to use **strings as case labels**?

- Will ask user to enter his day of birth(Monday,Tuesday,Wednesday,Thursday,Friday,Saturday,Sunday)
- But **switch expression/case label must need an integer**?
- We can do one thing, we can **ask user to enter integer values corresponding to days**
- Ex:
  - 0 for Monday
  - 1 for Tuesday
  - 2 for Wednesday etc

Enum

- Enum is **a user defined data type**
- Also known as **enumerated data type**
- Used **to assigns names to integral constants**, these names make a program easy to read and maintain

0-Monday
1-Tuesday
2-Wednesday
3-Thursday
4-Friday
5-Saturday
6-Sunday

Enum - Syntax

- enum enum-name {constant-1, constant-2, constant-3…constant-n}

```
enum days{Monday,Tuesday,Wednesday,Thursday,
Friday,Saturday,Sunday};
```

- By default, constant-1 is **0**, constant-2 is **1**, constant-3 is **2** etc..
- Next enumeration constants follow increment by 1

Enum declaration

- enum enum-name v1;
- enum enum-name v1,v2,v3;
- enum enum-name {

```
        constant-1,
        constant-2,
        constant-3…
        constant-n
        } v1;
```

```c
#include<stdio.h>
enum days{Monday=10,Tuesday,Wednesday,Thursday=50,
Friday,Saturday=20,Sunday};

int main(){
    enum days day=Monday;
    printf("%d\n",day);
    printf("Enter the day of your birth:0 for Monday, 1 for
Tuesday, 2 for Wednesday...6 for Sunday\n");
    scanf("%d",&day);
    switch(day){
case Monday:
    printf("magical");
    break;
case Tuesday:
    printf("terffic");
    break;
case Wednesday:
    printf("Wow");
    break;
case Thursday:
    printf("talented");
case Friday:
    printf("Fantastic");
    break;
case Saturday:
    printf("Smashing");
    break;
case Sunday:
    printf("Smily");
    break;
default:
    printf("nothing");
    break;
    }
    return 0;
}
```

Initializing values

```
enum days{Monday=10,Tuesday,Wednesday,Thursday=50,
Friday,Saturday=20,Sunday};
```

Monday – 10
Tuesday – 11
Wednesday – 12
Thursday – 50
Friday – 51
Saturday – 20
Sunday – 21

Use of Enum

- Used to **make large applications code readable and maintainable**
- used when we want our variable to have only **a set of values(Monday,Tuesday,Wednesday,Thursday,Friday,Saturday,Sunday)**

Valid expressions

- 2 + 3,
- 9 * 16 % 2,
- 10 / 2 + 5,
- ‘a’ ,
- ‘a’ + 1

```c
#include<stdio.h>
int main(){
    int num=1;
    switch(5+1){
    case 1:
        printf("one\n");
        break;
    case 3:
        printf("three\n");
        break;
    case 5:
        printf("five\n");
    case 2:
        printf("love\n");
    case 6:
        printf("zero\n");
    default:
        printf("welcome");
    }
    return 0;
}
```

```
zero
welcome
```

```c
#include<stdio.h>
int main(){
    int num=1;
    switch('a'>10){
    case 1:
        printf("one\n");
        break;
    case 3:
        printf("three\n");
        break;
    case 5:
        printf("five\n");
    case 2:
        printf("love\n");
    case 6:
        printf("zero\n");
    default:
        printf("welcome");
    }
    return 0;
}
```

```
one
```

```c
#include<stdio.h>
int main(){
    int num=1;
    switch(20.2){
    case 1:
        printf("one\n");
        break;
    case 3:
        printf("three\n");
        break;
    case 5:
        printf("five\n");
    case 2:
        printf("love\n");
    case 6:
        printf("zero\n");
    default:
        printf("welcome");
    }
    return 0;
}
```

Switch expression only supports Integral data types, Any other data types will throw an Invalid type error

```
error: switch quantity not an integer
```

```c
#include<stdio.h>
int main(){
    int num=1;
    switch(20){
    case 1:
        printf("one\n");
        break;
    case 20.0:
        printf("three\n");
        break;
    case 5:
        printf("five\n");
    case 2:
        printf("love\n");
    case 6:
        printf("zero\n");
    default:
        printf("welcome");
    }
    return 0;
}
```

The Constant expressions inside the case label must be integral. Any other datatype would throw an invalid type error

```
error: case label does not reduce to an integer constant
```

```c
#include<stdio.h>
int main(){
    int num=30;
    switch(num){
    case 10:
        printf("one\n");
        break;
    case 20:
        printf("three\n");
        break;
    case 5+5:
        printf("five\n");
    case 30:
        printf("love\n");
    case 60:
        printf("zero\n");
    default:
        printf("welcome");
    }
    return 0;
}
```

```
error: duplicate case value
```

Note

- **Only Integral values are allowed** inside the case label, and the **expression inside the switch must evaluate to a single integral constant**.

- Break and default are **optional**

- **Nesting is valid** for switch

Disadvantages of switch statement

- Supports **only Integral expressions/constants**.
- **Does not support more than one expression.**
- If **more expressions** need to be evaluated, **another switch** should be added and nested.

Can I write switch inside another switch?

```c
#include<stdio.h>
int main(){
    int num=30;
    switch(num){
    case 10:
        printf("one\n");
        break;
    case 30:
        printf("three\n");
        switch(num+10){
            case 30:
                printf("suresh");
                break;
            case 40:
                printf("john");
            case 50:
                printf("jerry");
                break;
            default:
                break;

        }
        break;
    case 40:
        printf("love\n");
    case 60:
        printf("zero\n");
    default:
        printf("welcome");
    }
    return 0;
}
```

```
three
johnjerry
```

15 levels of nesting

Difference between switch and else if ladder

Switch	If else
Easy to read and implement	When the number of cases is more, it is difficult to read and implement
Only Integral expressions are valid	Supports other datatype expressions/values as well
Fast compared to if-else	Slow compared to switch
If a matching case is found, all the statements following that case are evaluated till a break or end of switch is found.	Only one if/else-if block is executed and control jumps to the end of if..else if ladder.

Output of this program?

```c
#include<stdio.h>
void languages(){
    printf("c program\n");
    printf("java\n");
    printf("python\n");
}

int main(){
    printf("welcome\n");
    printf("suresh\n");
    printf("naresh\n");
    printf("hareesh\n");
    languages();
    printf("Thank you all");

    return 0;
}
```

```
welcome
suresh
naresh
hareesh
c program
java
python
Thank you all
```

```
welcome
c program
java
python
Thank you all
```

```c
#include<stdio.h>
void languages(){
    printf("c program\n");
    printf("java\n");
    printf("python\n");
}

int main(){
    printf("welcome\n");
    goto languageDetails;
    printf("suresh\n");
    printf("naresh\n");
    printf("hareesh\n");
    languageDetails:
    languages();
    printf("Thank you all");

    return 0;
}
```

```
welcome
c program
java
python
Thank you all
```

goto statement

- Used to **jump from one part of the code to any other part of the code**
- We can **alter the normal flow** of the program
- You can keep **goto** statement anywhere in the program

```c
#include<stdio.h>
void languages(){
    printf("c program\n");
    printf("java\n");
    printf("python\n");
}

int main(){
    printf("welcome\n");
    goto languageDetails;
    printf("suresh\n");
    printf("naresh\n");
    printf("hareesh\n");
    languageDetails:
    languages();
    printf("Thank you all");

    return 0;
}
```

You can keep goto statement anywhere

```c
#include<stdio.h>
void languages(){
    printf("c program\n");
    printf("java\n");
    printf("python\n");
}

int main(){
    printf("welcome\n");
    goto languageDetails;
    printf("suresh\n");
    printf("naresh\n");
    printf("hareesh\n");
    languageDetails:
    languages();
    printf("Thank you all");

    return 0;
}
```

```c
#include<stdio.h>
void languages(){
    printf("c program\n");
    printf("java\n");
    printf("python\n");
}

int main(){
    printf("welcome\n");
    printf("suresh\n");
    printf("naresh\n");
    printf("hareesh\n");
    languageDetails:
    languages();
    printf("Thank you all");
    goto languageDetails;

    return 0;
}
```

Be careful while writing backward jump

```c
#include<stdio.h>
void languages(){
    printf("c program\n");
    printf("java\n");
    printf("python\n");
}

int main(){
    char response;
    printf("welcome\n");
    printf("suresh\n");
    printf("naresh\n");
    printf("hareesh\n");
    languageDetails:
    languages();
    printf("Thank you all\n");
    printf("Have you completed above languages? Type y for yes, any other key for no");
    scanf("%c",&response);
    if(response=='y'){
        printf("very good");
    }else{
        goto languageDetails;
    }

    return 0;
}
```

- Because it creates an **infinite loop**
- You will need to write **proper conditions to come out of the loop**

You can keep goto statement anywhere

```c
#include<stdio.h>
void languages(){
    printf("c program\n");
    printf("java\n");
    printf("python\n");
}

int main(){
    printf("welcome\n");
    goto languageDetails;
    printf("suresh\n");
    printf("naresh\n");
    printf("hareesh\n");
    languageDetails:
    languages();
    printf("Thank you all");

    return 0;
}
```

Forward jump

```c
#include<stdio.h>
void languages(){
    printf("c program\n");
    printf("java\n");
    printf("python\n");
}

int main(){
    printf("welcome\n");
    printf("suresh\n");
    printf("naresh\n");
    printf("hareesh\n");
    languageDetails:
    languages();
    printf("Thank you all");
    goto languageDetails;

    return 0;
}
```

Backward jump

```
goto label;
--------------
--------------
--------------
label:
statements;
```

```
label:
statements;
--------------
--------------
--------------
goto label;
```

Note

- We should try to **avoid goto** as far possible to reduce the complexity and to improve the execution speed
- Loops(for, while, dowhile) were introduced for the better structure and readability

Scope, Life time and Visibility of the program

```c
#include<stdio.h>
int main(){
    int rank;
    printf("Welcome suresh\n");
    printf("What is your eamcet rank?");
    scanf("%d",&rank);
    if(rank<30000){
        printf("Very Good ");
    }else{
        printf("Kunchum baga chaduvu ");
    }
    printf("Your rank is : %d",rank);
    return 0;
}
```

```c
#include<stdio.h>
int main(){
    int rank;
    printf("Welcome suresh\n");
    printf("What is your eamcet rank?");
    scanf("%d",&rank);
    if(rank<30000){
        printf("Very Good ");
    }else{
        printf("Kunchum baga chaduvu ");
    }
    printRank();
    return 0;
}

void printRank(){
    printf("Your rank is : %d",rank);
}
```

```
Welcome suresh
What is your eamcet rank?30000
Kunchum baga chaduvu Your rank is : 30000
```

```
error: 'rank' undeclared (first use in this function)
```

What next?

- Storage classes