Introduction to Computing and Programming

Type Conversion, and Conditional Statements/Decision making in C, Loops

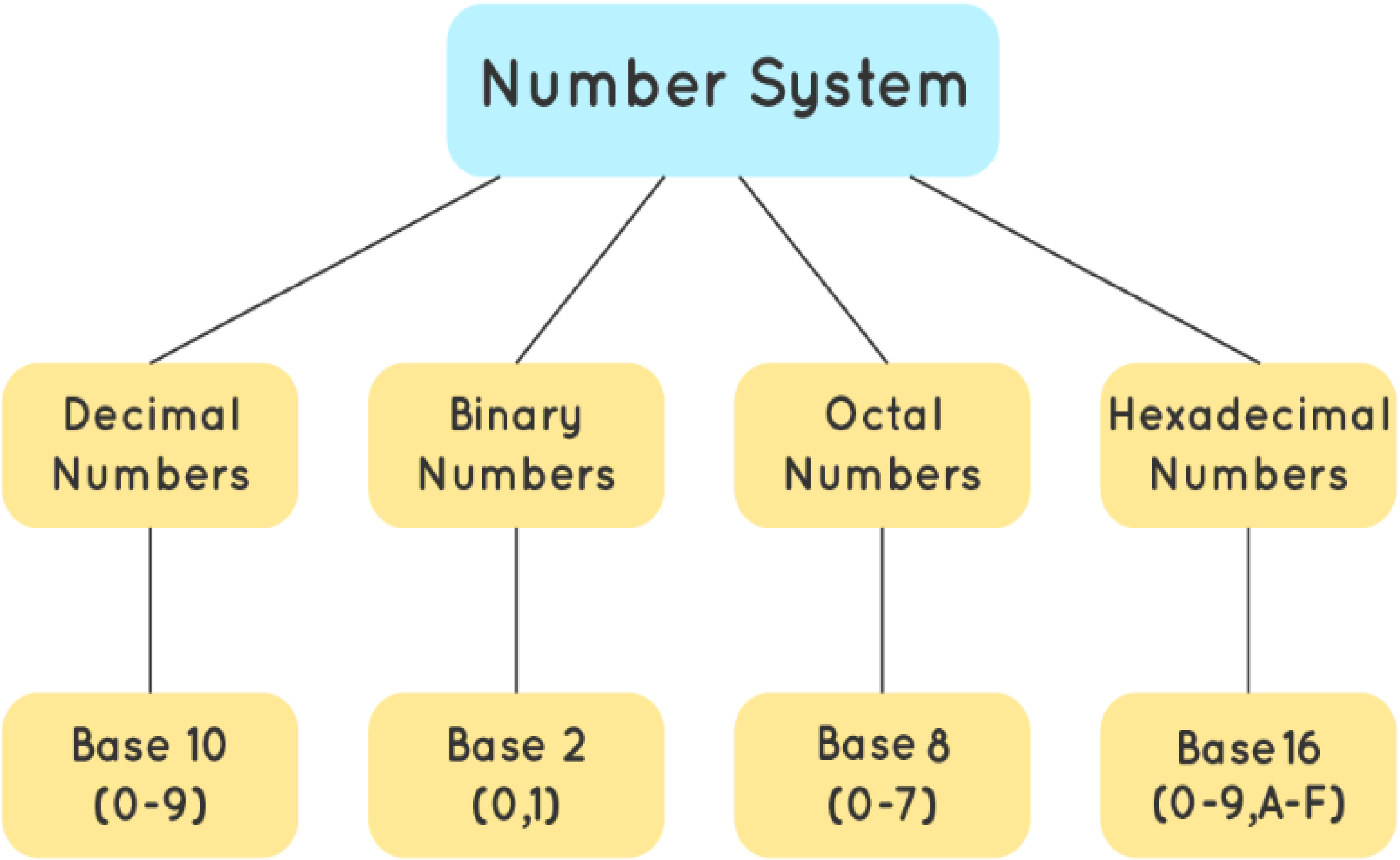
Recap

- NUMBER SYSTEM
- OPERATORS
- PRECEDENCE & ASSOCIATIVITY

Contents

- Example: Operators Precedence & Associativity
- Type Conversion
- Conditional Statements/Decision making in C
- Loops

Types of Number System



Number System

Decimal Numbers

Base 10
(0-9)

Binary Numbers

Base 2
(0,1)

Octal Numbers

Base 8
(0-7)

Hexadecimal Numbers

Base 16
(0-9,A-F)

Coding Question for Practice:

- **Write a C program to that takes input in integer but prints output in equivalent Octal & Hexadecimal.**

Hint: Use %o and %x specifier.

```
int main() {
   int number;

   // Input an integer from the user
   printf("Enter an integer: ");
   scanf("%d", &number);

   // Print the equivalent octal and
hexadecimal values
   printf("Octal equivalent: %o\n", number);
   printf("Hexadecimal equivalent: %X\n",
number);

   return 0;
}
```

Example

```
#include <stdio.h>
int main(void) {
  int x = 10, y = 5;
  y = x++ + ++y;
  printf("x = %d y = %d", x, y);
  /* post incr ++ is has highest priority, so x becomes 11 but it'll increase only after
the statement is evaluated, so it is not reflected in the value of 'y' y = 10 + 5 x = x +
1 */
  return 0;
}
```

11 $\cancel{\boxed{10}}$ $\cancel{\boxed{5}}$ 6 16

x

y = 10 + 6 = 16

++x + y++

Let’s Solve

10*4>>2 || 3

5/10*5+5*2

└→ 0*5 → 0+5*2

↓

10

5|10&12>>2

101

1010

1100 ⇒ 0011

1010

0011

0010

10/(5<10 && 20<30)

→ undefined (divide by 0 error)

10/(5-5)

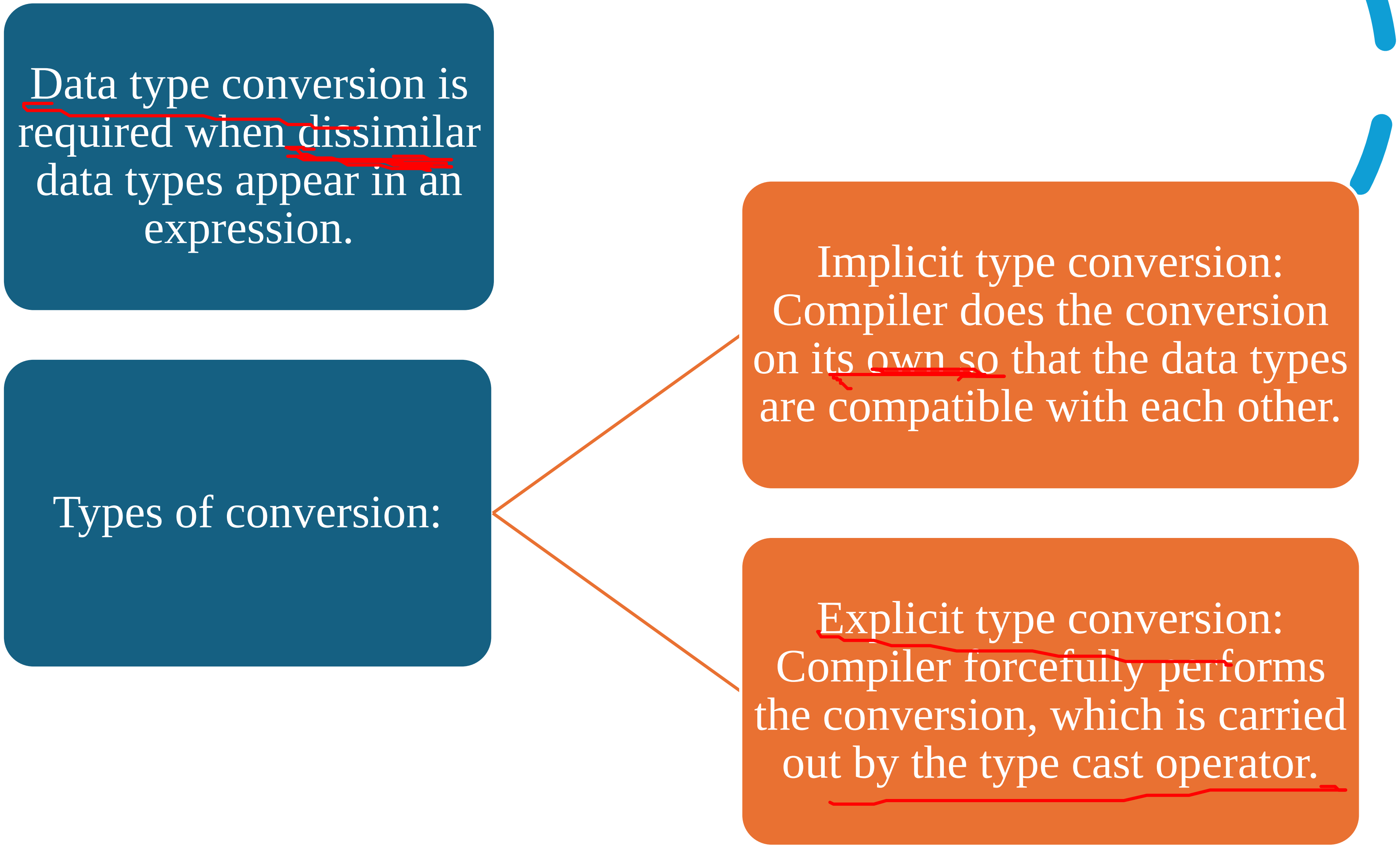
0010

0101

0111

└→ 7

Type Conversion in C



Data type conversion is required when dissimilar data types appear in an expression.

Types of conversion:

Implicit type conversion: Compiler does the conversion on its own so that the data types are compatible with each other.

Explicit type conversion: Compiler forcefully performs the conversion, which is carried out by the type cast operator.

Implicit type conversion

While performing implicit or automatic type conversions, the C compiler follows the rules of type promotions.

- Byte and short values: They are promoted to int.
- If one operand is a long: The entire expression is promoted to long.
- If one operand is a float: The entire expression is promoted to float.
- If any of the operands is double: The result is promoted to double.

Example

```
#include <stdio.h>
int main(){
    int  i = 17;
    char c = 'c';
    int sum;
    sum = i + c;
    printf(“Sum: %d\n", sum);
    return 0;
}
```

17+99

116

```
#include <stdio.h>
int main(){
    char a = 'A';
    float b = a + 5.5;
    printf("%f", b);
    return 0;
}
```

65

65+5.5

1.45

1.25

70.5

Explicit type conversion

C provides a typecast operator. You need to put the data type in parenthesis before the operand to be converted.

(int)

```
type2 var2 = (type1) var1;
```

It is only when type1 is greater in length than type2 that you should use the typecast operator.

Example

```
#include <stdio.h>
int main(){
   int x = 10, y = 4;
   float z = x/y;

   printf("%f", z);

   return 0;
}
```

2.000000

Output: 2.000000

```
#include <stdio.h>
int main(){
   int x = 10, y = 4;
   float z = (float) x/y;

   printf("%f", z);

   return 0;
}
```

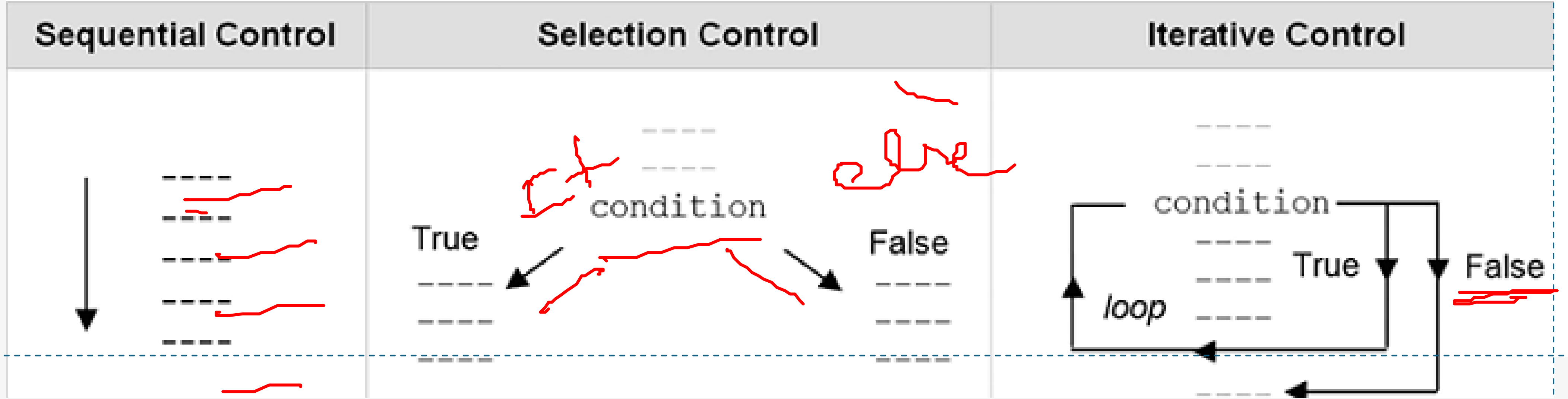
Output: 2.500000

Decision making in C

Control Structures

Control Structures

- Control flow is the order that instr~~uctions are executed in a~~ program.
- A control statement is a statement that determines the control flow of a set of instructions.
- Types of Control:
  - **Sequential control**: Instructions are executed in the order that they are written
  - **Selection control**: Selectively executes the instructions. **E.g.** Decision Control
  - **Iterative control**: Repeatedly executes the instructions. **E.g**. Loops.



Decision making statement

if Statement

if-else Statement

if-else-if Ladder

Nested if Statement

switch Statement

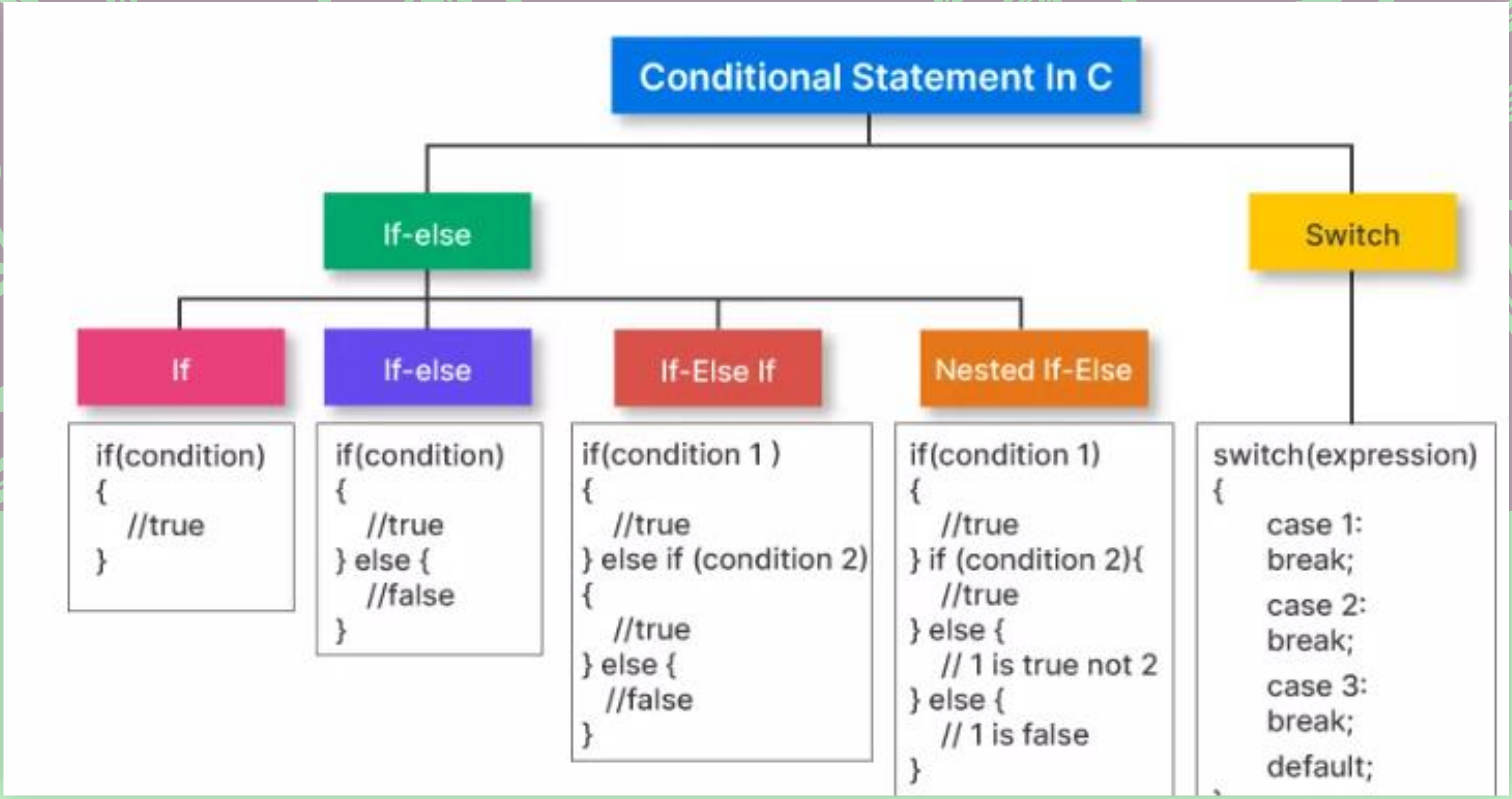
Conditional Operator

Jump Statements:

- Break
- continue
- goto
- return

a > b ? 10 : 15

20 30



Conditional Statement In C

```
if(condition)
{
    //true
}
```

```
if(condition)
{
    //true
} else {
    //false
}
```

```
if(condition 1 )
{
    //true
} else if (condition 2)
{
    //true
} else {
   //false
}
```

```
if(condition 1)
{
    //true
} if (condition 2){
    //true
} else {
    // 1 is true not 2
} else {
    // 1 is false
}
```

```
switch(expression)
{
      case 1:
      break;
      case 2:
      break;
      case 3:
      break;
      default;
```

If Statement

If(n%2==0)
Pf("even n");
else
Pf("odd n");

It is a selection control statement based on the value of a given Boolean expression

Expression’s value can be True or False.
We may want to do something only when a certain condition is true.

if statement takes an expression with it.

If the expression results to True

- then the block of statements under it is executed.

If it results to False

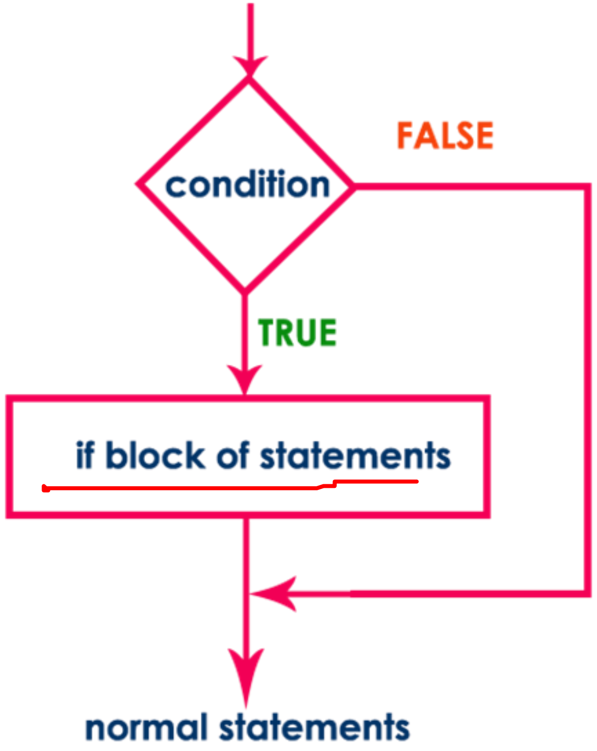
- then the block is skipped and control transfers to the statements after the block.

Syntax:

```
if (Boolean expression)
{
  statement(s)
}
```

Etc

Execution flow diagram



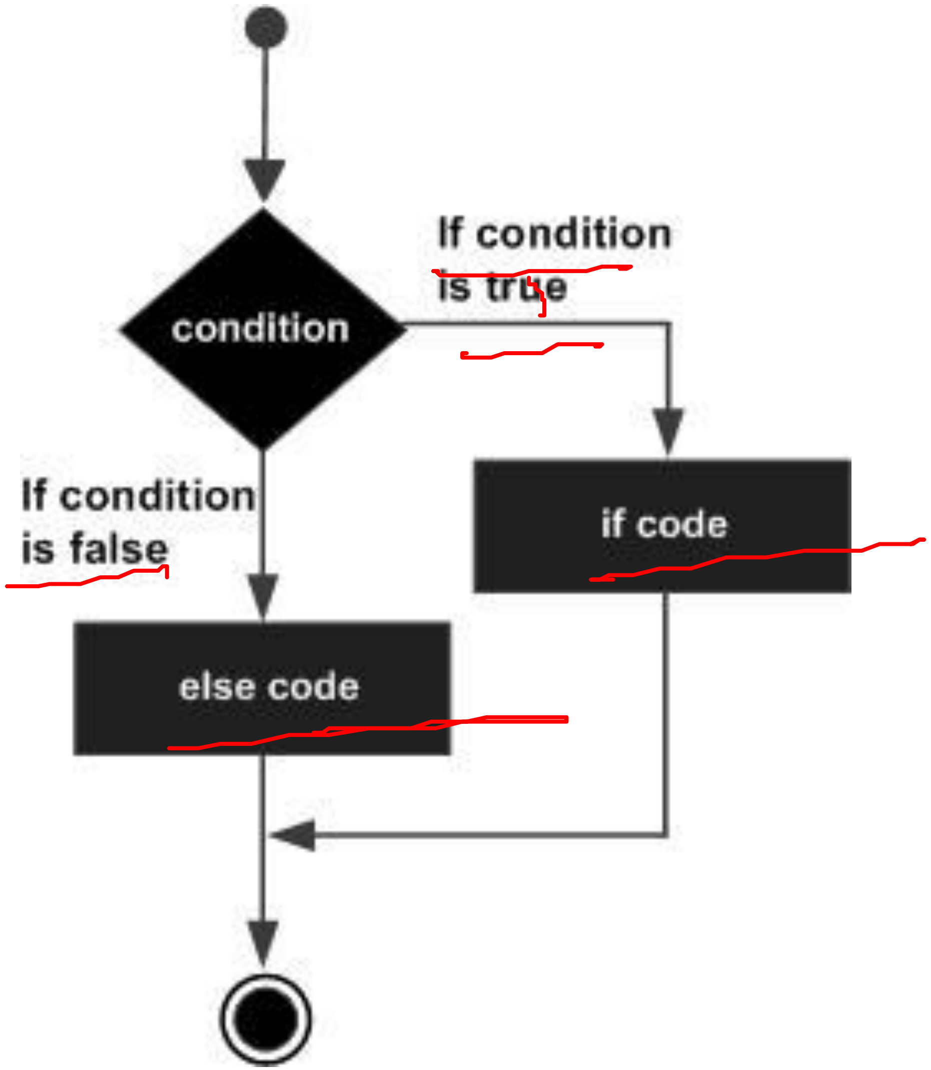
Example

```
#include <stdio.h>

int main()
{
float grade = 100;
if (grade == 100)
{
printf("You got a perfect grade");
}
}
```

```
#include <stdio.h>

int main()
{
int age = 18;
if (age >= 18)
{
printf("You are eligible for voting");
}
}
```

Over to you

Write a C program to check whether a number is positive or not.



If...else Statement

Syntax:

```
if (Boolean expr){
   statement;
   . . .
}
else{
   statement;
   . . .
}
```

If else statement

```
#include<stdio.h>
int main ()
{
  int num1, num2;
  num1=12;
  num2=13;
  if (num1 == num2){
    printf("both are equal");}
   else{
    printf(“Numbers are not equal”);}
  return 0;
}
```

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

If else if else statement

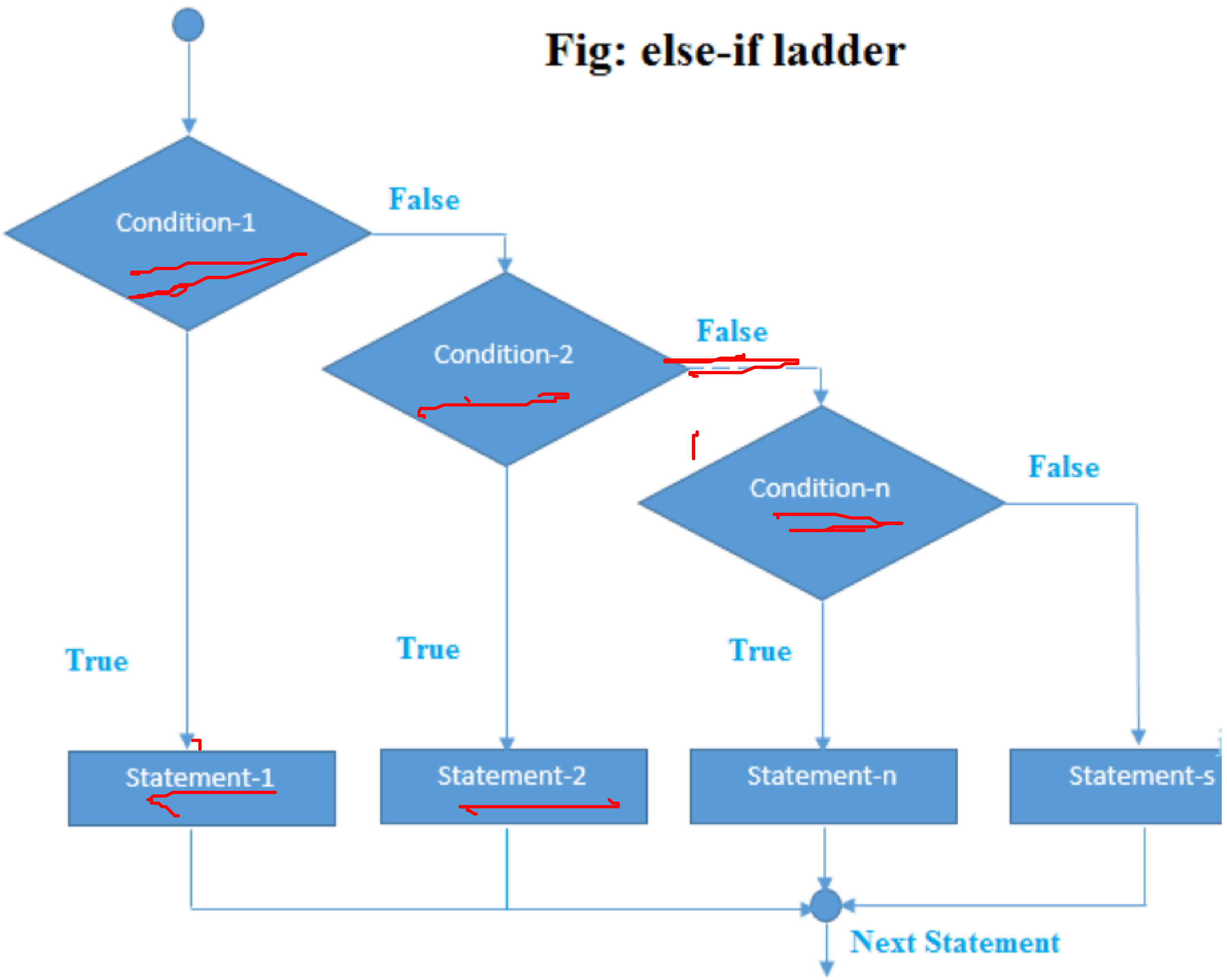


Fig: else-if ladder

Example

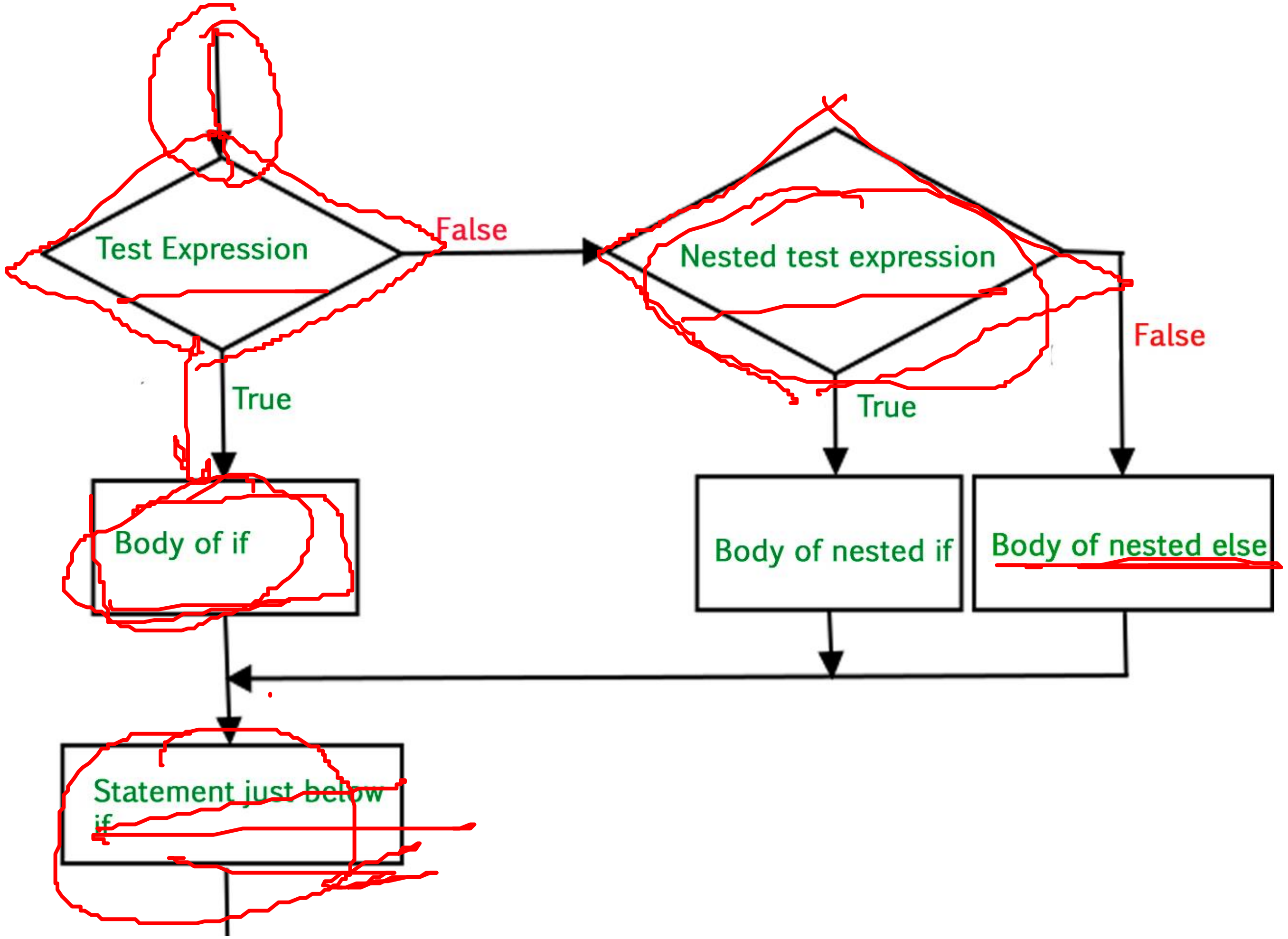
```
#include <stdio.h>
int main() {
    int n1, n2;
    printf("Enter two integers: ");
    scanf("%d %d", &n1, &n2);
    if(n1 == n2) {
        printf("Result: %d = %d",n1,n2);
    }
    else if (n1 > n2) {
        printf("Result: %d > %d", n1, n2);}
    else {
        printf("Result: %d < %d",n1, n2);
    }
}
```

```
#include<stdio.h>
int main()
{
int time;
printf("Enter current time value between 0-24:\t");
scanf("%d",&time);
if (time > 0 && time < 10) {
  printf("Good morning.");
}
else if (time > 10 && time < 20) {
  printf("Good day.");
}
else if(time > 20 && time < 24) {
  printf("Good night.");
}
else{printf("Enter valid time");}
}
```

Nested if statements (multi-way selection)

You can put an if statement in the block under ***another if statement***.

This is to implement further checks.



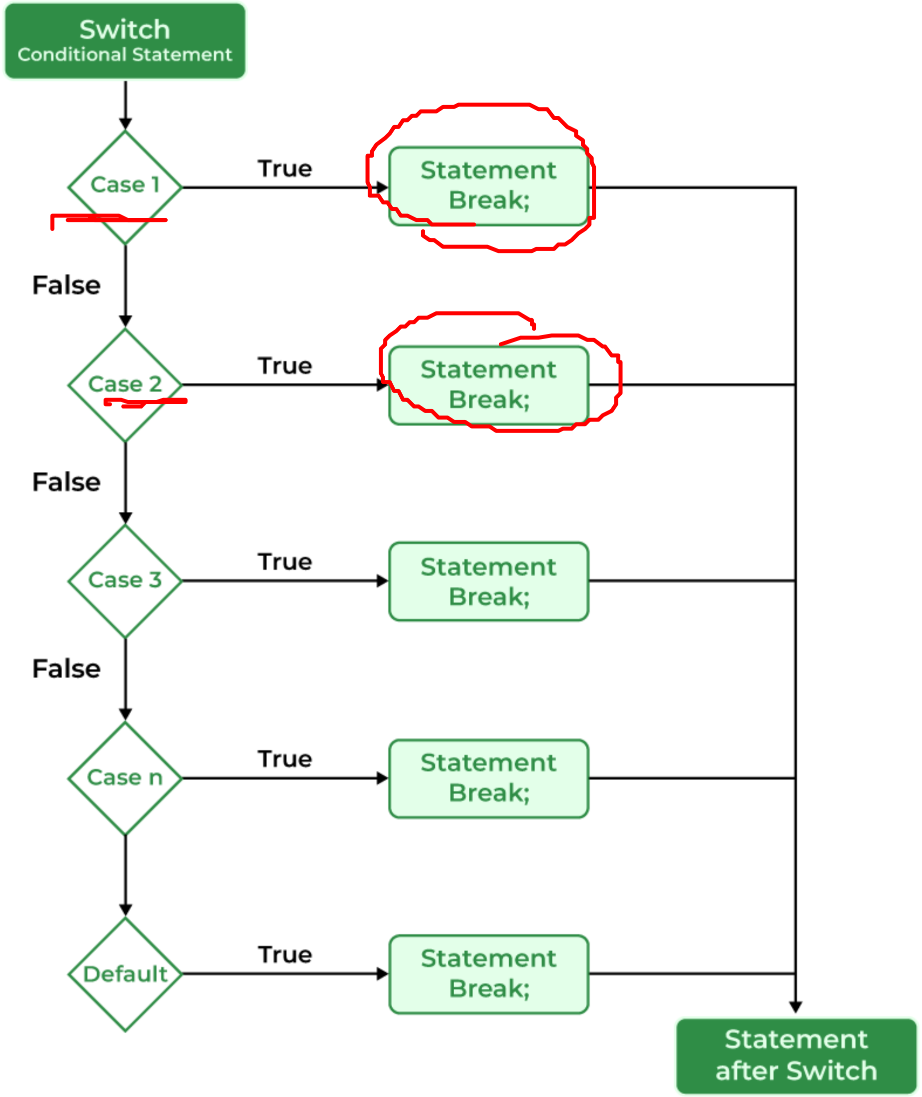
Nested if statement Example

```
#include <stdio.h>
int main() {
    int n1, n2;
    printf("Enter two integers: ");
    scanf("%d %d", &n1, &n2);
    if (n1 >= n2) {
      if (n1 == n2) {
        printf("Result: %d = %d",n1,n2);
      }
      else {
        printf("Result: %d > %d", n1, n2);} }
    else {
        printf("Result: %d < %d",n1, n2);
    }}
```

```
#include <stdio.h>
int main() {
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);
    if (num >= 0) {
      if (num == 0) {
        printf("Number is zero");
      }
      else {
        printf("%d is a positive number", num);
      }
    }
    else {
        printf("%d is a negative number", num);
    }
}
```

Switch case statement

- Evaluates a given expression and based on the evaluated value(matching a certain condition), it executes the statements associated with it.
- Switch case statements follow a selection-control mechanism.
- It is a substitute for long if_statements that compare a variable to several integral values.
- The switch statement is a multiway branch statement.



Rules of the switch case statement

The “**case value**” must be of “**char**” and “**int**” type.

There can be one or N number of cases.

The values in the case must be **unique**.

Each statement of the case can have a break statement. It is optional.

The default Statement is also optional.

How switch Statement Work?

Step 1: The switch variable is evaluated.

Step 2: The evaluated value is matched against all the present cases.

Step 3A: If the matching case value is found, the associated code is executed.

Step 3B: If the matching code is not found, then the default case is executed if present.

Step 4A: If the break keyword is present in the case, then program control breaks out of the switch statement.

Step 4B: If the break keyword is not present, then all the cases after the matching case are executed.

Step 5: Statements after the switch statement are executed.

Switch case example

```
#include <stdio.h>
int main()
{  int var = 2;
   switch (var)
   {
   case 1:
      printf("Case 1 is executed.\n");
   case 2:
      printf("Case 2 is executed.\n");
   case 3:
      printf("Case 3 is executed.\n");
   case 4:
      printf("Case 4 is executed.");
   }return 0;
}
```

5/3

Switch case with break keyword

```
#include <stdio.h>
int main() {
  int day = 4;
  switch (day) {
    case 1:
      printf("Monday");
      break;
    case 2:
      printf("Tuesday");
      break;
    case 3:
      printf("Wednesday");
      break;
    case 4:
    printf("Thursday");
      break;
    case 5:
      printf("Friday");
      break;
    case 6:
      printf("Saturday");
      break;
    case 7:
      printf("Sunday");
      break;
  }
   return 0;
  }
```

The break statement is unconditional exit from the switch case or loop

Syntax of break in C

```
break;
```

Switch case with break and default keyword

```
#include <stdio.h>
int main() {
  int day = 9;
  switch (day) {
    case 1:
      printf("Monday");
      break;
    case 2:
      printf("Tuesday");
      break;
    default:
      printf(“No case matched");
    }
  return 0;
}
```

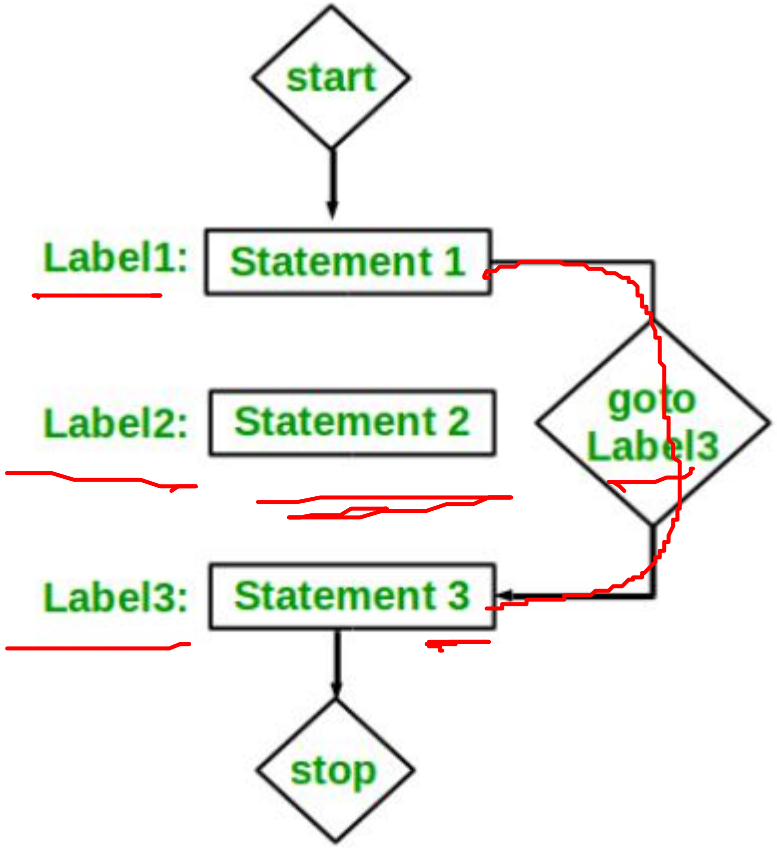
The default statement is used to run some code if there is no case matches

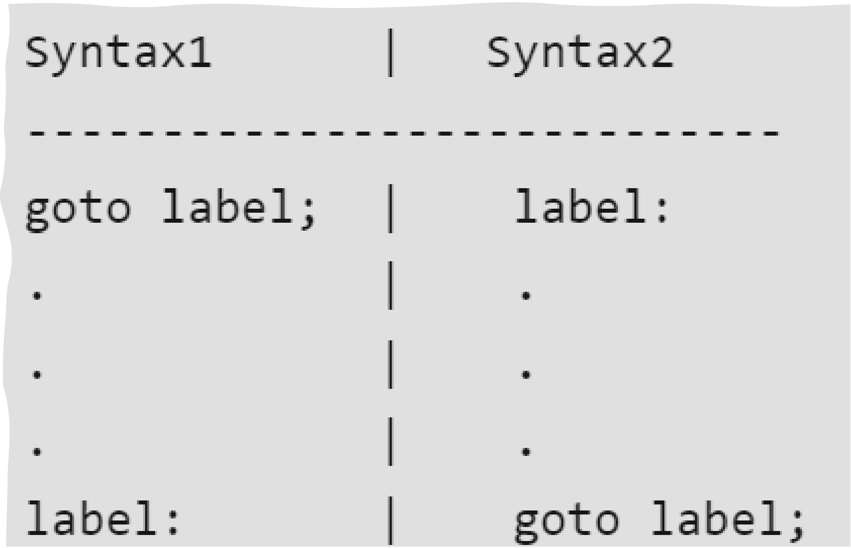
Jump statement : goto

- The C **goto statement** is a jump statement
- Referred as an **unconditional jump** statement.
- The goto statement can be used to jump from anywhere to anywhere within a function.

- The first line tells the compiler to go to or jump to the statement marked as a label.

- The ‘label:’ can also appear before the ‘goto label;’

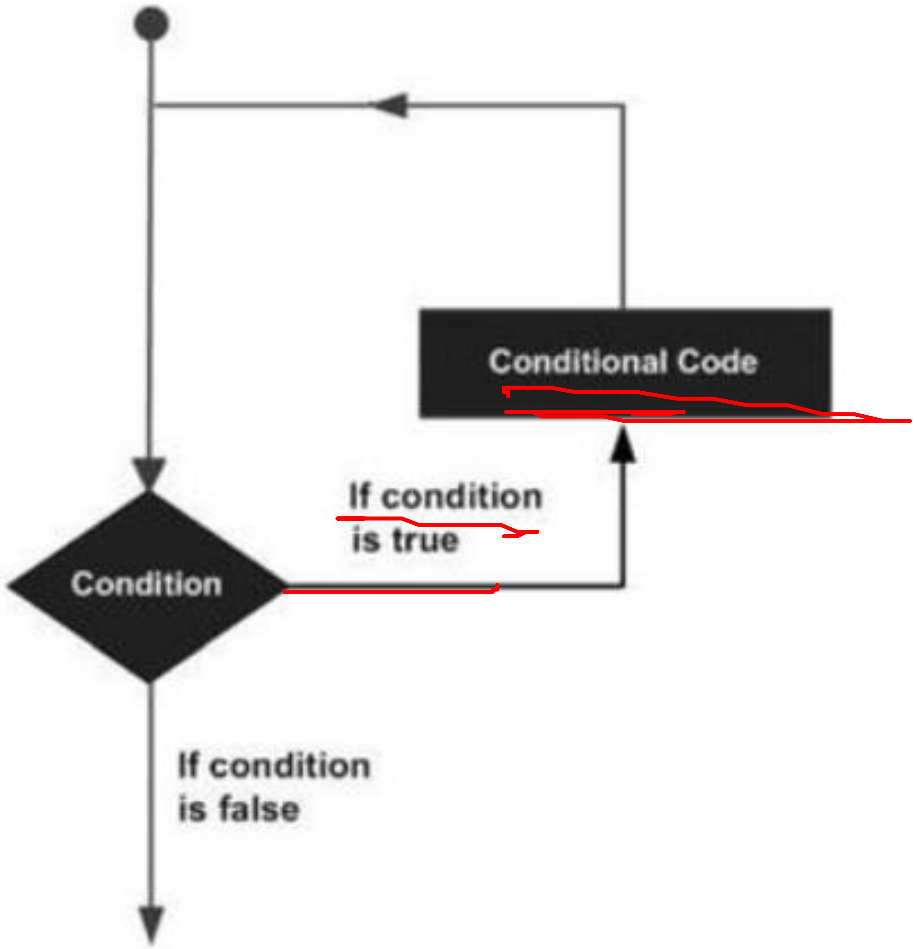




```
Syntax1         |   Syntax2
----------------------------
goto label;     |    label:
.               |    .
.               |    .
.               |    .
label:          |    goto label;
```

Jump statement: goto Example

```
#include <stdio.h>
int main()
{
int num = 25;
if (num % 2 == 0) {goto even;}
else {goto odd;}
even:
   printf("%d is even", num);
   return 0;
odd:
   printf("%d is odd", num);
   return 0;
}
```



What is Loop?

- A loop is a sequence of instructions that is **continually repeated** until a certain condition is reached.
- They reduce the need for repetitive coding and **improve efficiency**.

Types of Loop

- There are **three types** of loops:
  - Using a while statement
  - Using a for statement
  - Using a do-while statement

Upcoming lecture

Loops & its Exercise