Date ___/___/_____ Saathi

C ***

1st step → Preprocessing
Step 2 → Compilation
Step 3 → Assembly
step 4 → Linking
Step 5 → Loading

Program: Keywords, Identifiers, constants, string literal, symbols
Is made of

☆ Variables & Datatypes

Variables

* A name given to a memory location
* Declared by writing type variable_name;
  int a;
  int a, b;
* Initialized & declared by type variable_name = value;
  int a = 4;

☆ Defining a variable

** Can contain alphabets, digits, underscore (_)
  int harry; int _harry;
* A variable name can start with an alphabet and underscore only. int 9harry; ✗
* Can't start with a digit
* No whitespace & reserved keywords is allowed.
  int harry shubham; ✗

Page No.

Date ___/___/_____ Saathi

* Valid variable names: int harry, float harry123, char _harry3
* Invalid variable names: $harry, int 7harry, char long.

☆ Data types

* Basic Data Type: int, char, float, double.
* Derived Data Type:- array, pointer, structure, union
* Enumeration Data type:- enum
* Void Data type - void.

☆ Basic operators & functions

+ - * /
int a;
printf("%d", 3+7);
scanf("%d", &a);

functions

sizeof ()

Page No.

Saathi

Date ___/___/_____

★ Operators

* Is a symbol used to perform operations on given programming language

★ Types

* Arithmetic op. Eg:- + - * / %
* Relational op. Eg:-
  - == equal to
  - != not equal to
  - > Greater than
  - < less than
  - >= Greater than or equal to
  - <= less " " " "

* Logical op. Eg:- &&, || → 1 || 0 ⇒ true

  (1) && (1) ⇒ true

  or

  ! →

* Bitwise op. ⇒ 2 & 3

  (10)
  & (11)

  10

  0 → 00
  1 → 01
  2 → 10
  3 → 11

a	b	a & b	a | b	a^b
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Saathi

Date ___/___/_____

Other bitwise op.

~ → Binary one's complement op.
<< → Binary left shift operator
>> → Binary right shift op.

* Assignment op. Eg:- = &, int a=7;
  - += (Add AND assignment) a+=1; ⇒ a=8
  - -= " "
  - *= " "
  - /= " "

* Miscellaneous op.

sizeof → Returns the size of a variable
& → " address of " "
* → Pointer to a variable
?: → Conditional expression

★ Operator Precedence

* Associativity of an operator is a property that determines how operators of the same precedence are grouped in the absence of parantheses.

Q1 Print a multiplication table of a no. entered by the user in pretty form

Eg:- Input → Enter the no. you want multiplication table of
6
output → Table of 6.

★ Format specifiers, constants, Escape sequence.

★ Format specifier → It is a way to tell the compiler, which type of data type we are using.

Syntax→ int a=8;
float b=7.333;
printf("The val. of a is %d and the value of b is %f");

★ Eg: printf("Hello is a good boy %a.2f", var);
will print var with 2 decimal points on a 'a' character space

Syntax printf("The val of a is %d and val. of b is %f.4f", a, b)

★ Imp format specifiers

Sy
1 %c → character
2 %d → integer
3 %f → floating point no.
4 %l → long
5 %lf → double
6 %Lf → long double

Page No.

★ Constants

Constant → Can Cannot be changed

★ There are 2 ways to define a Constant in C

1 Const keyword
2 #define preprocessor

Syntax→ int a=8;
Const float b=7.333;
b=7.22 //wrong, since b is a Constant

★ Escape Sequence → It is a sequence of characters. It doesn't represent itself when used inside string literal or character.

(\n) → Single character

\t → Tab (Horizontal)
\n → New line
\b → Backspace
\a → Alarm or Beep
\' → Single Quote
\" → Double Quote

Page No.

Saathi

Date ___ / ___ / ____

★ Comments in C

Eg Virtual Assistant. C

1100 lines

```
Int func1() {
    return //returning the value
}
/* Compiler will ignore
   */
```

★ C if else statements → Control statements

→ It's used to perform operation based on some conditions

Types of if statements

1- if statement
2- if else statement
3- if else if "
4 Nested if "

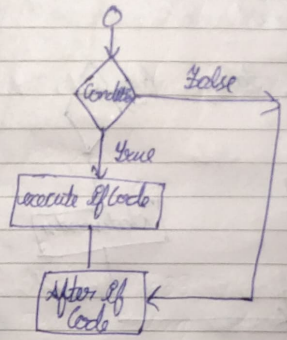
★ Syntax for if-else

```
if (3>2) {
    printf("It's will execute");
}
```

Saathi

Date ___ / ___ / ____

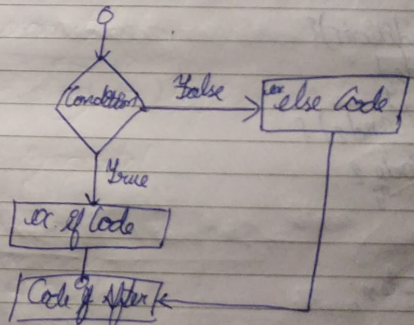
★ Flow chart for if statement



Syntax

```
if (Condition) {
    Code;
}
```

★ Flowchart for if else statement



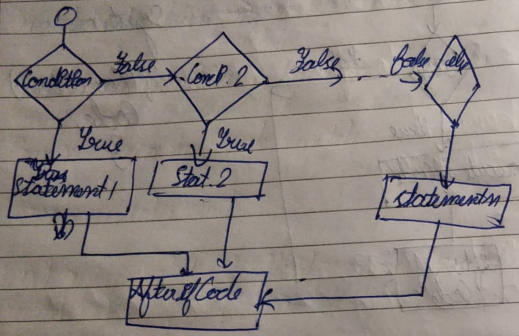
Syntax

```
if (Condition) {
    if true
}
else {
    if false
}
```

Saathi

Date ___/___/_____

★ Flowchart of if-else if ladder



Syntax -

```
if (condition 1){
    Code 1
}
else if (Cond 2) {
    Code 2
}
.
.
else {Codeₙ;
}
```

Page No.

Saathi

Date ___/___/_____

eg:-

```
printf ("Enter your age\n");
scanf ("%d", &age);
printf ("You have entered %d as your age\n", age);
if (age > 18) {
    printf ("You can vote");
}
else {
    printf ("You cannot vote");
}
return 0;
```

Q Maths Science - 45
Science - 15
Maths - 15

★ Switch Case Statements

Syntax → switch (a) {

```
case 2:
    printf ("Value is 2");
    break;
case 3:
    printf ("Value is 3");
default:
    printf ("Nothing matched");
```

Page No.

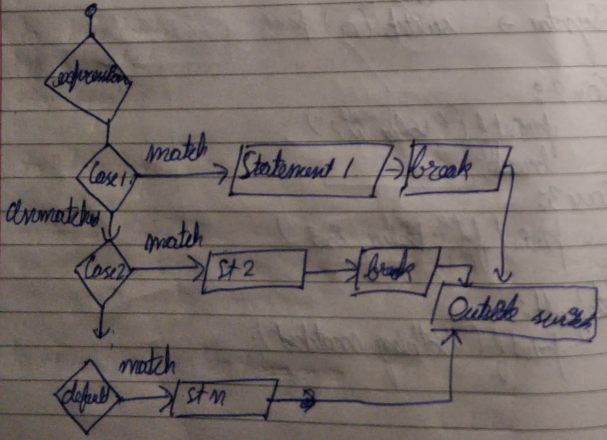
Date ___/___/_____

★ Rules for switch statement

1. Switch expression → must be int or char
2. Case value must be integer or character
3. Case must come inside switch
4. break is not a must

Eg:- `int x;` Valid switch | `float x` Invalid switch
`switch(x) {`
`// Cases + Code`
`}` ✓ | `switch`

★ Flowchart for switch case statement



Eg:-
```
printf("Enter your age\n");
scanf("%d", &age);
switch(age)
{
case 3:
    printf("The age is 3");
    break;
case 13:
    printf("The age is 13");
    break;
case 23:
    printf("The age is 23");
    break;
default:
    printf("Age is not 3, 13, 23");
    break;
}
```

Nested switch

Syntax:
```
switch()
{
case :
    switch()
    {
    }
}
```

```
printf("Enter your marks");
scanf("%d", &mark");
switch(age)
{
case 3:
    printf("The age is 3\n");
    switch(marks)
    {
    case 45:
        printf("Your marks are 45");
        break;
    }
```

★ Loops

* Advantages of using loops

1 Code Reusability
2 Saves Time
3 Traversing

index=0;
[loop start
printf (index)
inc. index
loop end]
~~unt~~ untill cond.
fails to satisfy.

★ Basic Syntax

int i=0;
Loop starts → Check the condition —false→ exit the loop
i<10 ↑
↓ true
execute loop
i=i+1;

* Types of loops

1. do while loop.
2. while loop.
3. for loop.

1 do while loop

Syntax →
```
do{
  //code to be executed
} while (Condition)
```

```
int i =0;
do {
i=i+1;
printf ("%d",i);
} while (i<10);
```

* do while executes atleast ~~one~~ once.

i	0
i	1

Eg:-
```
int num, index=0;
printf("Enter a number:\n");
scanf("%d", & num);
do
{
 printf("%d\n", index);
  index = index +1;
} while(index<num);
return0;
```

★ While loops

Syntax →
```
while (Condition) {
  // Code to be executed
}
```

Eg:-
```
int i=0;
while (i< 30){
printf ("%d\n", i);
i = i+1;
}
```

Eg:-

Saathi

Date __/__/____

* For loops

1. The for loop is used to iterate the statements or a part of the program several times.
2. It is used to traverse the data structures like the arrays and linked lists.
3. It has a little different syntax than while and do while loops.

Syntax →

```
for(expression1; exp2; exp3){
    //code to be run
}
```

```
int i;
```

Eg:-
```
for(i=0; i<5; i++){
printf("%d", i);
}
```

* Properties of exp1:

1. Initialization of the loop variable
2. We can initialize more than one variable in exp1.
3. exp1 is optional

Page No.

Saathi

Date __/__/____

Eg:-
```
int i;
for(i=0; i<5; i++)
{
printf("%d", i);
}
return 0;
```

```
int i, j;
for(i=0, j=0; i<5; i++)
{
printf("%d %d\n", i, j);
}
```

```
int i=0, j=0;
for(; i<5; i++)
{
//
}
```

* Properties of exp2

1. Checks for specific conditions to be satisfied
2. It can have more than one condition however, the loop will iterate until the last condition becomes false.
3. It is optional
4. Can perform the task of exp1 & exp3

Page No.

Date ___/___/___

☆ Properties of exp3

1. It is used to update the loop variable.
2. We can update more than 1 variable at a same time.
3. It is optional

```
int i, j = 0;
for(i = 0; i<5; )
{
  printf("%d%d \n", i, j);
  i++; j++;
}
```

☆ Break & Continue

1. Used to bring the program control out of the loop

```
while (Cond.) {
//user enters his name & keeps playing until
gameover
  if (name=="Shubham") {
      break;
  }
```

```
int i;
for (i=0; i<10; i++){
  printf("%d\n", i);
}
return 0
```

Date ___/___/___

```
int i, age;
for(i=0; i<10; i++){
  printf("%d \n Enter your age", i);
  scanf("%d", &age);
  if (age>10)
  {
   break;
  }
```

2 The break statement is used inside loops or switch statement
3. Break statements can be used with.
(i) loops
(ii) Switchcase expression

☆ Continue

1. Used to bring the program control to the next iteration of the loop.

2. The statement

```
while (Cond.) {
    if (name == "harry") {
        Continue;
    }
    {1000 lines
     S1
}
```

2 Continue statement skips some code inside the loop and continues with the next iteration.

3. It skip some lines of code for a particular condition

Date ___/___/_____

Syntax

```
int i, age;
for(i=0; i<10; i++){
printf("%d \n enter your age \n", i);
scanf("%d", &age);
if (age > 10)
{ continue; }
printf("We have not come across any continue statement");
                ||
                ||
                ||
                ||
printf("Harry is a ~~get~~ good boy");
}
return 0
```

★ Goto statement :-

1. Also called jump statement
2. Use to transfer program control to a predefined label.

Syntax :-

```
label: ↓
  .
  .
goto label
```


Date ___/___/_____

3. Its use is avoided since it causes confusion for the fellow programmers in understanding the code
4. Is preferable when we need to break multiple loops using a single statement.

Syntax

```
label:
  printf("We are inside label"); → goto end;
goto label;
end:
printf("We are at end");
}
return 0;
```

Example

```
for (int i=0; i<8; i++)
{
printf("%d", i);
for (int j=0; j<8; j++)
{
printf("\n Enter the number, enter 0 to exit \n");
scanf("%d", &num);
if (num == 0){
    break; → goto end
}

end
```

Saathi

Date ___/___/_____

☆ Typecasting

```
int a=3;
printf("The value of a is %d\n",a);

float b=54;
printf("The value of b is %d\n",(int)b);

float b=(float)54/5;
```

☆ Function

1. Used to divide a large C problem into small pieces
2. Function can be called multiple times to provide reusability and modularity to the program
3. Also called procedure or subroutine

Syntax:-

```
return_type function_name(data_type para1, data_type para
{
  //code
}
int printstar() {
   printf("*");
   return0;
}
```

Page No.

Saathi

Date ___/___/_____

* Advantages of function

1. We can avoid rewriting same logic through function
2. We can divide work among programmers using function
3. We can easily debug a program using function

* Declaration, Definition & Call

1. A function is declared to tell a compiler about its existence

```
int printstar();
    main() {
    }
```

2. A function is defined to get some task done

```
int printstar() {
   printf---   ] actual implementation
   ---
   return0;
}
```

3. A function is called in order to be used

```
int printstar();
   int main() {
      printstar(); → function call
   }
```

Page No.

Saathi

Date

Types of function

1. Library functions - functions included in C header files.
2. User defined functions - function created by (programmer to reduce complexity of a program.

```
int sum(int a, int b);
```

Eg:- 1. Without arguments & without return value.
2. Without arguments and with return value.
3. With arguments & without return value
4. With arguments & with return value

```
int sum(int a, int b){
                  ① ②
    return a+b;        ③
}
main
int a=1, b=2, c;
c = sum(a, b);
```

Eg:- (Tutorial)

```
int sum(int a, int b)   // Declaration of function
{
return a+b;
}
int main()
{
int a, b, c;
a=7;
b=87;
c=sum(a,b);
printf("The sum is %d\n", c);
}
```

Eg:- ③ With argument, without return value ①

```
void printstar(int n)
{
for(int i=0; i<n; i++)
{
 printf("%c", '*');
}
int main()
{
printstar(7);
}
```

② Without argument, with return value ④

```
int takenumber()
{
 int i;
printf("Enter a number");
scanf("%d", &i);
return i;
}
int main()
{ int c;
c= takenumber();
printf("The number entered is %d\n", c);
return 0;
}
```

Date ___/___/_____

Saathi

① Without argument, without return value

```
void print();
{
int a;
printf("Enter a number:");
scanf("%d", &a);
}
the main()
printf("The number entered is", a);
}
```

Recursion

1. Is a process when a function calls a copy of itself to work on a smaller problem
2. Any function which calls itself is called recursive function
3. It makes the life of programmer easy by dividing a given problem into easier problem
4. Termination condition is imposed to stop them executing copies of themselves forever.
5. Any problem that can be solved recursively, can also be solved iteratively.

Date ___/___/_____

Saathi