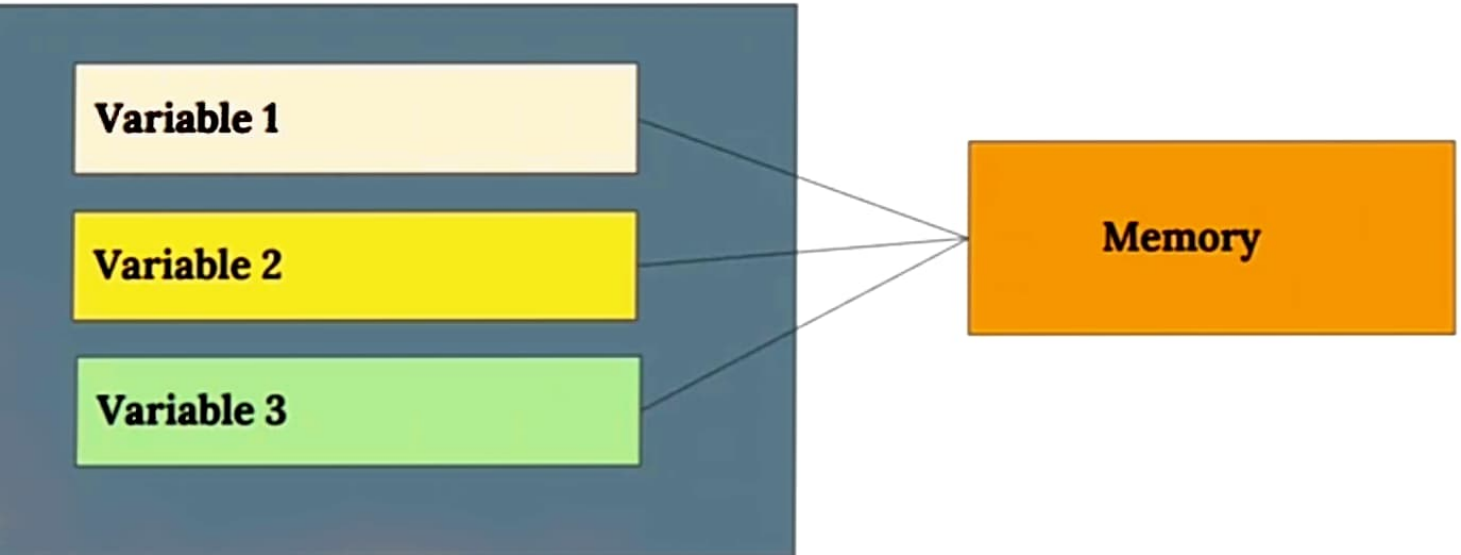
data types.

variables.

share the same memory location



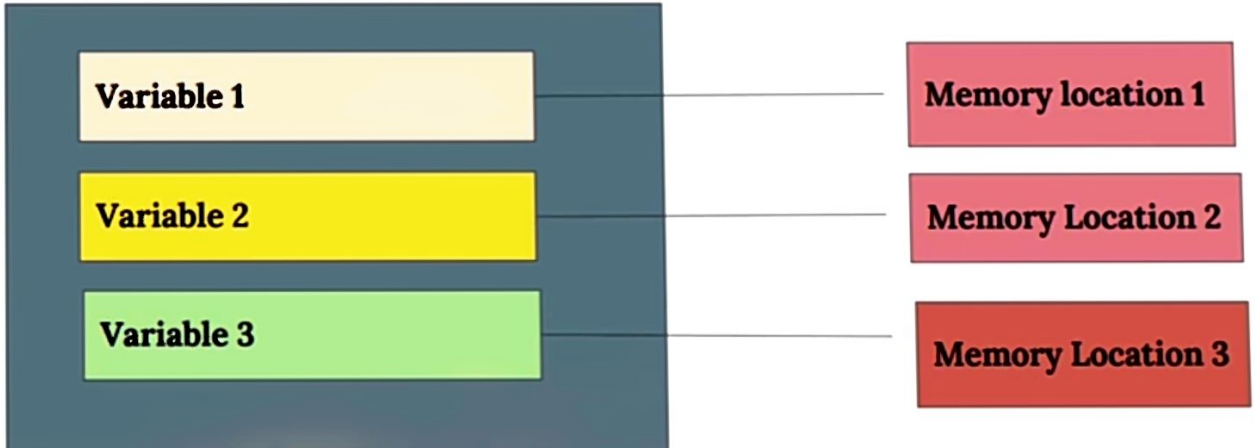
```
struct Person
{
   char name[50];
   int citNo;
   float salary;
};

int main()
{
   struct Person person1, person2, p[20];
   return 0;
}
```

Structures

Structures are user defined data types.

They are a collection of variables.

Structure allocates space for each variable separately



Boolean

The Boolean data type is used to declare a variable whose value will be set as true (1) or false (0).

To declare such a value, you use the bool keyword.

The variable can then be initialized with the starting value.

A Boolean constant is used to check the state of a variable, an expression, or a function, as true or false.

```c
#include<stdio.h>
#include<conio.h>
#include<math.h>
int main(){
   short int a = 100;
   int b = -1000;
   long int c = 1300;
   float d= 230.47;
   double e = 200.347;
   printf("sqrt(si): %d \n",sqrt(a));
   printf("pow(li, 3): %d \n",pow(b, 3));
   printf("sin(d): %d \n",sin(c));
   printf("abs(i) : %d \n",abs(d));
   printf("floor(d): %d \n",floor(e));
   printf("sqrt(f): %d \n",sqrt(d));
   printf("pow(d, 2): %d \n",pow(d, 2));
   return 0;
}
```

Math Library

C supports large number of mathematical functions.

These functions are for various mathematical calculations.

These functions can be directly used in programs.

We need to include <math.h> to use these functions which is a C library

String functions are accessible by importing "string.h"

Function	
strlen()	find length of string
strcpy()	copy one string to other
strcat()	join two strings
strcmp()	compare two strings
strlwr()	converts string to lowercase
strupr()	converts string to uppercase

Strings

greatlearning
Learning for Life

Strings are used to store sequence of characters as information.Strings can be used by importing the "*string*" library.

- **Concatenation**

Pointers in C

Pointers are variables which store the address of a variable.

They have data type just like variables, for example an integer type pointer can hold the address of an integer variable and an character type pointer can hold the address of char variable.

Syntax:

```
int a =10;

int *p=&a;
```

Definition:

An array can be defined in the following manner in C++

Data_type <name_of_array>[<size_of_array>]

Size of an array must be an integer as it shows the number of elements in an array which cannot be a real number

Syntax:

int a[10];//single dimensional array

double b[10][10]; //double - dimensional array

Array

An array is a collection of elements of similar data type which are stored in contiguous memory locations.

Every element in an array has a specific index.

Size of array in C is fixed at the time of definition.

Array's are of two types:

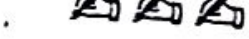
1) Single dimensional
2) Multi-dimensional

[1x4]

[2 X 3]

Long Question Answers:

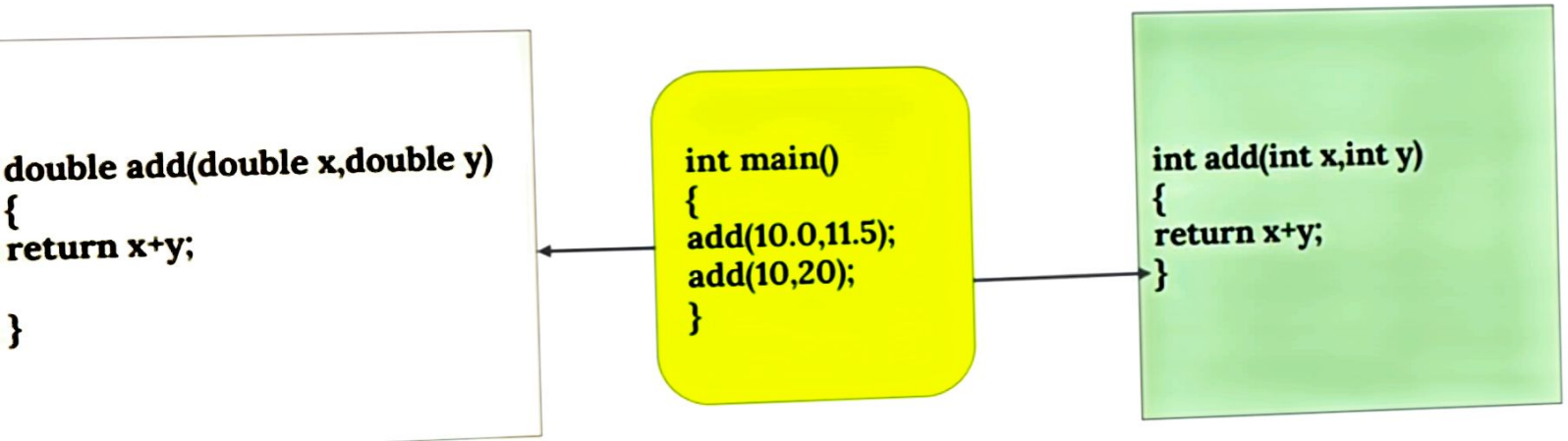
1. Define ecosystem. Explain structure of ecosystem.
2. Write a note on functions of ecosystem.
3. Explain with diagram the flow of energy and nutrient cycling seen in open system.
4. Explain - Energy flow is the 'flow of energy through living things within an ecosystem.'
5. Explain the difference between food chain and food web.
6. Write a note on different types of food chain in details.
7. Write in brief about ecological succession and its types.
8. Explain different types of ecosystem in details.
9. Explain different terrestrial ecosystem and write any one related case study in details.
10. Explain different aquatic ecosystem and write in details any one case study for aquatic ecosystem in detail.



Function Overloading

unction Overloading is the process of having two or more functions with same name but ifferent function definitions.

he return type, type of parameters or the number of parameters which the function takes ust be different for function overloading to happen.



Functions

Functions are blocks of code which are used to perform specific tasks .

In C a function needs to be declared before it's used.

Functions have a function definition , function body and a return type.

Functions with return type not as void need to return a value at the end.

Functions with return type void do not return any value.

```
#include <stdio.h>
int main() {
   int e = 20;
   switch (e) {
   case 10: printf("10");
      break;
   case 20: printf("20");
      break;
   default: // will be executed if e doesn't match any cases
      break;
}
return 0;
}
```

Switch case

The switch statement works as a multiway branch statement.

Modified form of if-else

It's a common alternative to the if statement when you want to get multiple results.

Default condition gets executed when no conditions are met.

- **The if Statement:**

  Selection and execution of statement(s) based on a given condition is done by the if statement. If the condition evaluates to True then a given set of statement(s) is executed. However, if the condition evaluates to False, then the given set of statements is skipped and the program control passes to the statement following the if statement.

  Syntax:

```
if (condition)
{
statement 1;
statement 2;
statement 3;
}
```

The if - else statement causes one of the two possible statement(s) to execute, depending upon the outcome of the condition.

```
if (condition)   // if part
{
Statement1;
statement2;
}
else      // else part
Statement 3;
```

Here, the if-else statement comprises two parts, namely, if and else. If the condition is True the if part is executed. However, if the condition is False, the else part is executed.

A nested if-else statement contains one or more if-else statements. The if else can be nested in three different ways, which are discussed here.

- The if - else statement is nested within the if part.

```
if (condition)
{
Statement l;
if(condition2)
Statement 2;
else
Statement 3;
}
else
Statement 4;
```

Conditional Statements

Conditional statements are also known as selection statements.

They are used to make decisions based on a given condition.

If the condition evaluates to True, a set of statements is executed, otherwise another set of statements is executed.

Conditional statements evaluate to a boolean value.

```c
#include<stdio.h>
int main() {
   int a;
   float b;
   double c;
   char d;

   // sizeof evaluates the size of a variable
   printf("Size of int: %ld bytes\n", sizeof(a));
   printf("Size of float: %ld bytes\n", sizeof(b));
   printf("Size of double: %ld bytes\n", sizeof(c));
   printf("Size of char: %ld byte\n", sizeof(d));

   return 0;
}
```

4. **Bitwise operator**

- used to perform bit-level operations on the operands.
- operators are first converted to bit-level and then the calculation is performed on the operands.
- mathematical operations such as addition, subtraction, multiplication etc. can be performed at bit-level for faster processing

5. Assignment Operator

- **Used for value assignment to a variable.**
- **Left side operand of the assignment operator is a variable and right side operand of the assignment operator is a value.**
- **The value on the right side must be of the same data-type of variable on the left side otherwise the compiler will raise an error.**
- **Example- += -= *= /=**

2. **Relational Operator**

- Used for comparison of the values of two operands.
- For example, checking for equality of operands, whether an operand is greater than the other operand or not etc.
- Some of the relational operators are (==, >= , <=)

3. **Logical Operator**

- used to combine two or more conditions/constraints
- result of the operation of a logical operator is a boolean value either true or false.
- Some logical operators are (AND(&&),OR(||),NOT(!))

Operators

C supports primarily 5 types of operators :

1. **Arithmetic Operators**

   Used for mathematical and arithmetic operators

   There are two types of arithmetic operators:

- **Unary Operators - Operators that operates or works with a single operand are unary operators. For example: (++ , - -)**
- **Binary Operators - Operators that operates or works with two operands are binary operators. For example: (+ , – , * , /)**

Format Strings

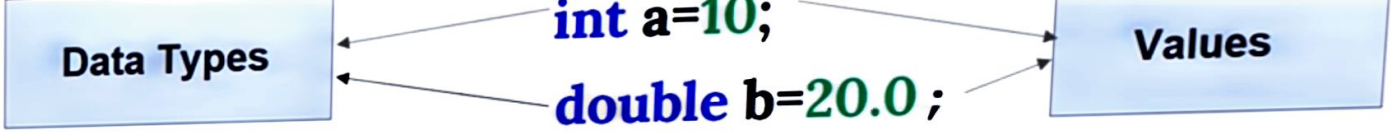
This informs the scanf() function, what type of input to expect and in printf() it is use heads up to the compiler, what type of output to expect.

%d	integer
%f	floating point
%c	character
%s	string

User Input/Output

C supports the following functions for user input/ output.

- printf() and scanf()
- getchar() and putchar()
- gets() puts()




```
#include <stdio.h>
#include <conio.h>
int main()
{
   int a=10;
   double b=20.0;

   printf("%d + %lf = %lf",a,b,a+b);
   return 0;
   getch();
```

What?
Memory references whose value changes during execution.

Why?
Makes memory handling easy.
Make reuse of same memory locations for storing different values.
Makes referencing of values easier.

Syntax!!

Initialization
Data_type [variable_name];

Assignment
[variable_name]=value;

C Data Types