Date.
Page.

Several Paradigm of Programming

Procedural

C, Pascal, Basic, fortran

Functional

LISP

OOP

C++, Java, Smalltalk

Rule or logic base

Flow.

Need of so many: → Choice of paradigms & thus language depends on how humans best think about the problem.

- → Efficiency
- → Compatibility of existing code.
- → Availability of translators.

Elementary data types

Declaration

i) Numerical data

Scalar - it represents a single object that has only one value
Composite - files, pointers, strings

Numerical data types
- i) integers
- ii) floating point real numbers
- iii) fixed point real number

Other data types
Complex number
Rational numbers
Boolean
Characters

Integers data types
- arithmetic
- Relational
- Logical

Floating point
- To solve round off issues

eg:- $10.5 = 0.105 \times 10^{-2}$

Mantissa exponent

Fixed point data types

- Real number with predefined decimal places.

eg: number (5,3)
↳ precision

Complex numbers

Two storage location.

Imaginary part Real Part

Rational numbers

Represented as $\frac{p}{q}$ and $q \neq 0$

Boolean

True = 1, False = 0

Own

Implimentation:

Composite data types:

i) Characters Strings

- concatenation of two strings
- Relational operation (=, <=, > etc).
- Dynamic string (evaluated at run time).

Fixed declare length: String longer than specified length are truncated.

iv) Type checking

1 Q. Reasons for studing concepts of programing language?

2 Q. Header files with angular braces (< >) and double quote (" ") ?

3 Q. What is pointer to pointer variable?

4 Q. How the negative integer is stored in memory.

5 Q. How records and ~~arrays~~ arrays are different?

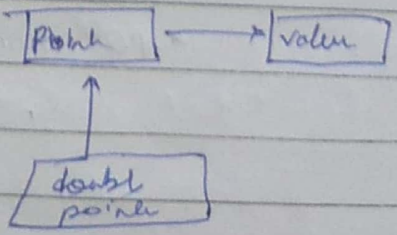
6 Q. How structure and array are different.

2 Ans: Header files with (< >) angular braces are predefined in language library and header files defined in double quotes are defined with a path.

~~Header~~ For '< >' compiler searches for the file in the include path list but for double quotes (" ") it searches for the file in current directory and then in include list.

3 Ans: A pointer stores the address of a variable, while a double pointer is used to store the address of a pointer.

```
int v=20;
int *p=&v;
int **q=&p;
```



Date:
Page:

4 Ans: Integers are stored in memory using two's complement format with the sign bit set to 1.

5 Ans: Record :- A data structure that stores bits of data of multiple types.

Record has 'record set'.

(i) It is possible to input or output data into the fields of an entity to create a record using records. (It is a data structure that consists of groups of data based records).

~~(ii) Ans~~

Arrays: it is possible to input or output data or store an array using for loop.

ii) Record can be used to store a collection of different data types

iii) Equivalent to a row or a tuple in a table.

ii) Used to store data types

iii) Arrays have location which are number. Every location in an array is of same data type.

Program / Syntax of record in C language.

```
struct employee
{
  int id;
  int age;
  float salary;
  char name[20];
};
```

Storage representation

ID →	1020
Age →	25
Salary →	21000
Name →	Raj

eg

Record

Representation in memory (Records)

i) It depends upon fields that are stored in ~~conjuctive~~ conjuctive positions in the same order as they are declared in record.

eg:- data could be stored as a record containing
numeric ~~age~~ year field
month field represented as string.

ii) Record can exist in any storage medium including main memory & mass storage devices.

a) Implement vector bounds exam.

Date.
Page.

Attributes of records

i) No. of components
ii) Data type of each component
iii) Records are sometimes are called structures.

Associative Arrays:-

Within an associative array we can pair information in a way that associates the key to the value, so that are getting more information to us than an indexed array.

→ It can hold other datatypes as keys other than
→ It helps to store & access string.
→ It gives easy to access & use facility.

'⇒' is also called this operator

$x = array (65, 46, 85, 48, 64)

Index
Array 0 = 65
1 = 46
2 = 85
3 = 48

```
<?php
$x = array ("divesh" => 65,
            "Aman" => 46);
```

key	value
first name	Kris
Last name	glomart

Date.
Page.

Keys	Values
0 —	Kris
1 —	Glomart
2 —	81
3 —	21000
4 —	2.5

$employee = array
("first name" ⇒ "Kris"
"Last name" ⇒ "glomart"
"Salary" ⇒ "21000"
"Age" ⇒ "81"
"Duration of service" ⇒ "2.5"

Index

i) Integers begins at zero

ii) It is used when you identify things by their position

Associative

i) They have strings have keys and behave like two column table.

ii)

Javascript eg:

```
Var y = ["Blue", "Green", "Red"]
y[0] = Blue
y[1] = Green
y[2] = Red
```

Date.
Page.

```
Var z = { color : "Blue"
          fabric : "Silk"
          size : "M" }
```

We call them object in Js

In Perl - 3 item associative array 1st member = key
2nd member = values

```
%, Employee = ("Michel", 'A',
               "Donnis", 'B',
               "Mone", 'D');
```

Assignment

Date: 21/2/19
Page:

1 Q. Show a situation during the execution when a data object of that type does not exist that is neither a variable nor a constant.

2 Q. Given eg. of primitive op^v that has
- a) an implicit argument.
- b) undefined for some data objects

3 Q. Give an example of op^v in programming lang.
- a) that is implemented directly in hardware
- b) that is implemented as subprogram

Q. How subprograms are defined?

Ans:- - Subprograms are abstract op^v defined by programmers.
- There are two type of op^v
  - i) specification of subprogram
  - ii) implementation of subprogram

i) a) name of function
b) & prototype
also called signature.
arguments / parameters

ii) The action performed by the subprogram.

•) In assembly lang. MACROS work as subprograms & programs.

Date.
Page.

Q. Modular approach of programming?

Ans. ↳ program divided into modules.
↳ unit testing is done of individual modules.
regression testing
alpha testing
beta testing

- makes error checking & debugging easy.
- testing is made easy.

→ implementation

```
int fun(int a, int b)
{ if(a>b)
    return a;
  else
    return b;
}
```

Data Abstraction

We define abstract data types as

i) Set of data objects.
ii) Set of abstract op's on those data objects.

eg:- class Ab
{ int a;
public:
void get data (int x) { a=x; }
int put data () { return a; }
virtual void fun () = 0;
};

Q:) See real life example of abstract class

Coupling

It is the degree of intra dependencies of different or more than two subprogram.

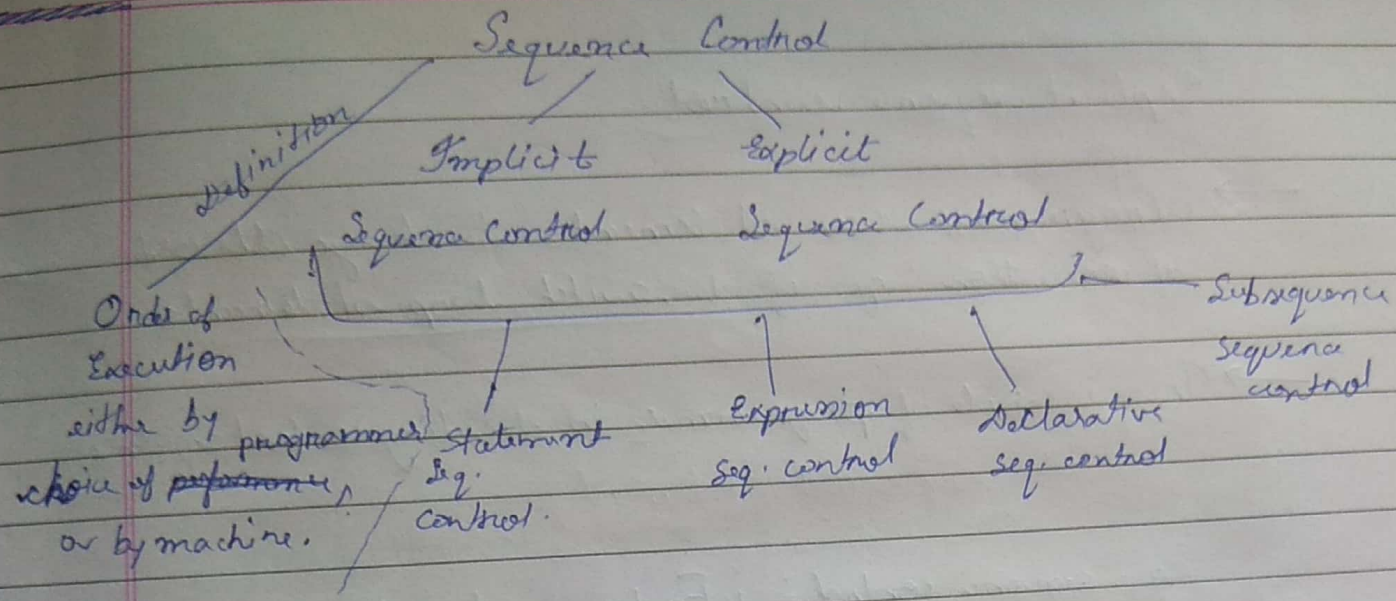
i.e how closely interconnected they are.

Cohesion

Individuality of subprogram.

:) Modular approach says low coupling & high cohesion

Sequence Control



Sequence control:-

It refers to the users action and computer logic, that initiates interrupt and terminate the transaction.

It is defined as the control of the order of execution of operation both primitive & user defined.

Implicit seq. control:-

It is determined by the order of the statement of the source program.

⇒ No role of programmer.
⇒ All work are done by programming language.
⇒ Compiler or any other translator of any programming that automates the sequence to solve the expression.

eg:- a+b*c

operator's precedence known by compilers.

eg:- Vaccum control, Washing Machine

Date.
Page.

Explicit sequence control

→ Programmer uses statements to change the order of execution eg:- if-else, switch, loop statements.

→ This is role of programmer in deciding control sequence.

eg:- sequence control :- i) Factory automation
ii) Industrial machines

a) device that operates - switches
b) devices that actuates machines - motors, valves
c) device that show the status of machines - lights, alarms (lamp, buzzer)

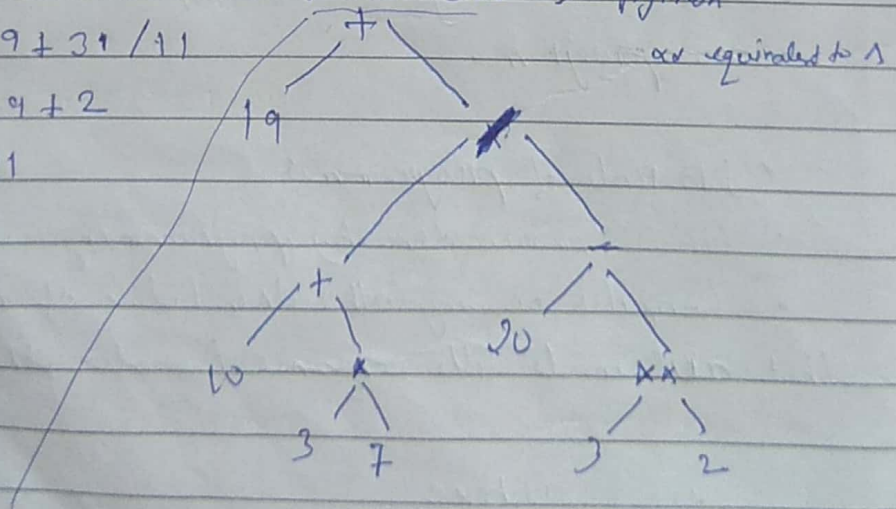
Sequence Control in expression:-

eg: 19 + (10 + 3 * 7) / (20 - 3 ** 2) python

= 19 + 31 / 11

= 19 + 2

= 21



** equivalent to ^

Tree

Date.
Page.

Sequence control in statements

i) Alteration
ii) Iteration
iii) Composition.

Iteration:- i) for loop ii) while loop iii) do while.

```
for(initialisation, condition, updation)
{
    ==== block of statements.
};
```

↓true

↓false (out of block)

If condition true
i) execution
ii) updation
iii) condition

```
while (condition)
{
}
```

↓true

↓false

out of block

```
do
{
}while(condition);
```

↑true

↓false

out of block.

→ one time execution

→ check condition

Alterations

i) if condition
ii) case statement
+ goto, continue & break.

Composition

↳ compound statement.

eg:- simple block - sequential flow

Declarative sequence control:

↳ no sequence is there

eg:- Prolog (programming & logics) language have a database ~~data~~ & structures containing facts & figures & rules.

Once we put a query, these facts & figures are analysed from the data base & answer is made accordingly.

Subsequence seq. control

↳ defines ~~the~~ role of subprograms.

Q. Diff. bet^n subprograms & subroutine & function.

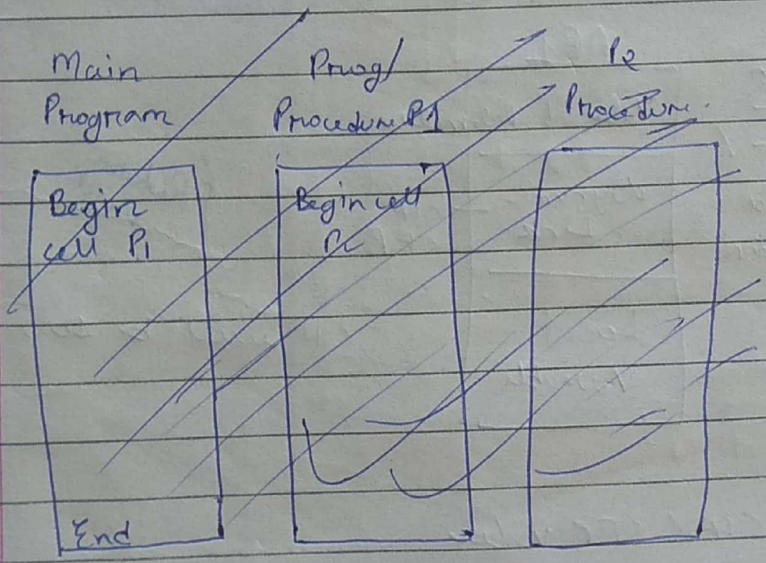
Date.
Page.

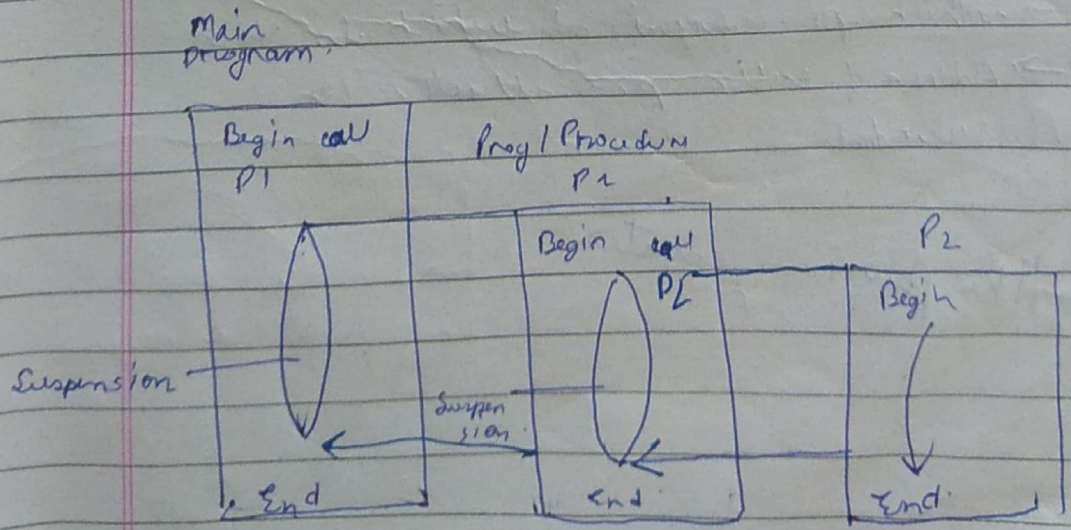
Subsequence or Subprogram Control

i) Simple call return subprogram
ii) Recursive subprogram

Subsequence / subprogram control
- Simple call return subprogram
- Recursive subprogram

Activation record
- CIP → Current Instruction pointer
- CEP → Current Environment pointer





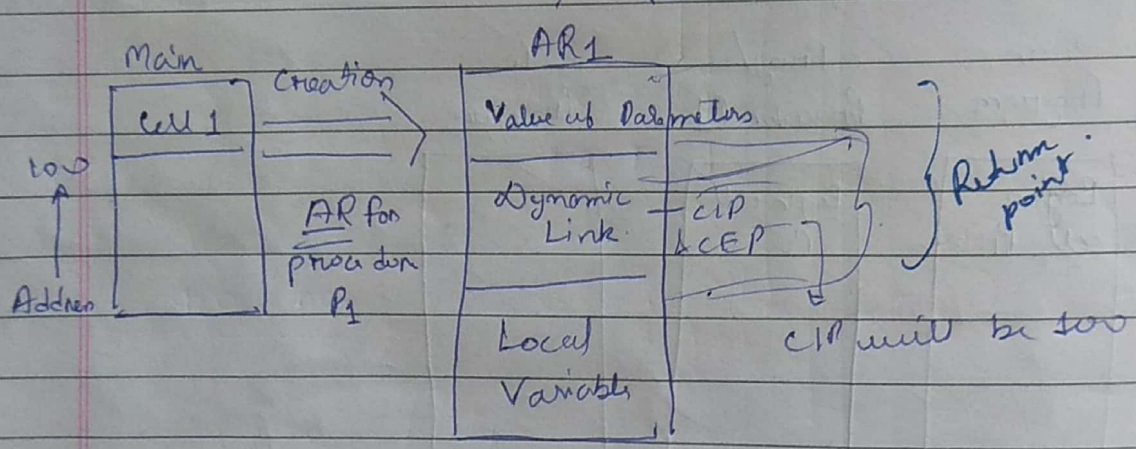
CEP & CIP = dynamic link

Date.
Page.

c) CEP stores the address of activation record.

Main Prog
↓
Call P_1
↓
Activation record is active
↓ Main prog suspended. until P_1 is ended
P_2 called under P_1 & P_1 is suspended.

Implementation:



Q. See qns on CIP & CEP values.

CIP → which procedure is at which location.
CEP → Address of all activation records.

like program counter

Recursive Subprogram

Date.
Page.

Subprogram
↓
Procedures that calls / encapsulates another program → Executing a subprogram is AR
↓
activation record.

AR
|
1. CIP
2. CEP → Program counter

Ex. Subprogram B from subprogram A

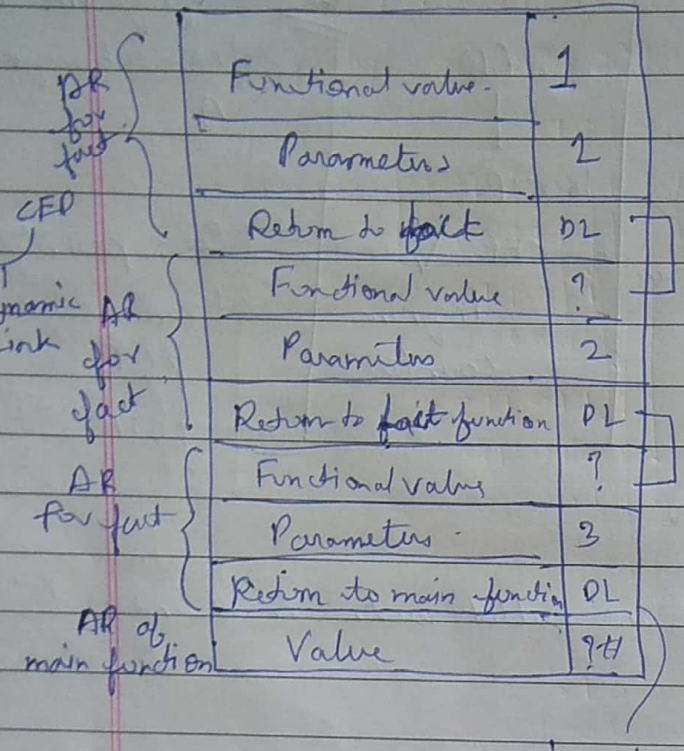
→ Exe points at an address of B
→ CIP

*) when a subprogram is called activation record is created.

Calling program — CIP (address of A)
— CEP (address of B).
↳ contains.

Location of current instruction is called CEP.
Location of instruction pointer of B is called CIP.

also called
prog. counter
location counter
or instruction counter.




```
int fact(int n)
{
  if (n<=1) return 1;
  else
  return (n * fact(n-1));
}
void main()
{
  int value;
  value = fact(3); //calling
  print(" ");
}
```

dynamic link

put async keyword in python

activation record in stack

Coroutines (cooperative routines (function))

Concurrent programming

yield keyword in python & JS

↳ function that can pause & resume its execution

Implementation of Sub-program

1) Actual code 2) Non-coding part.

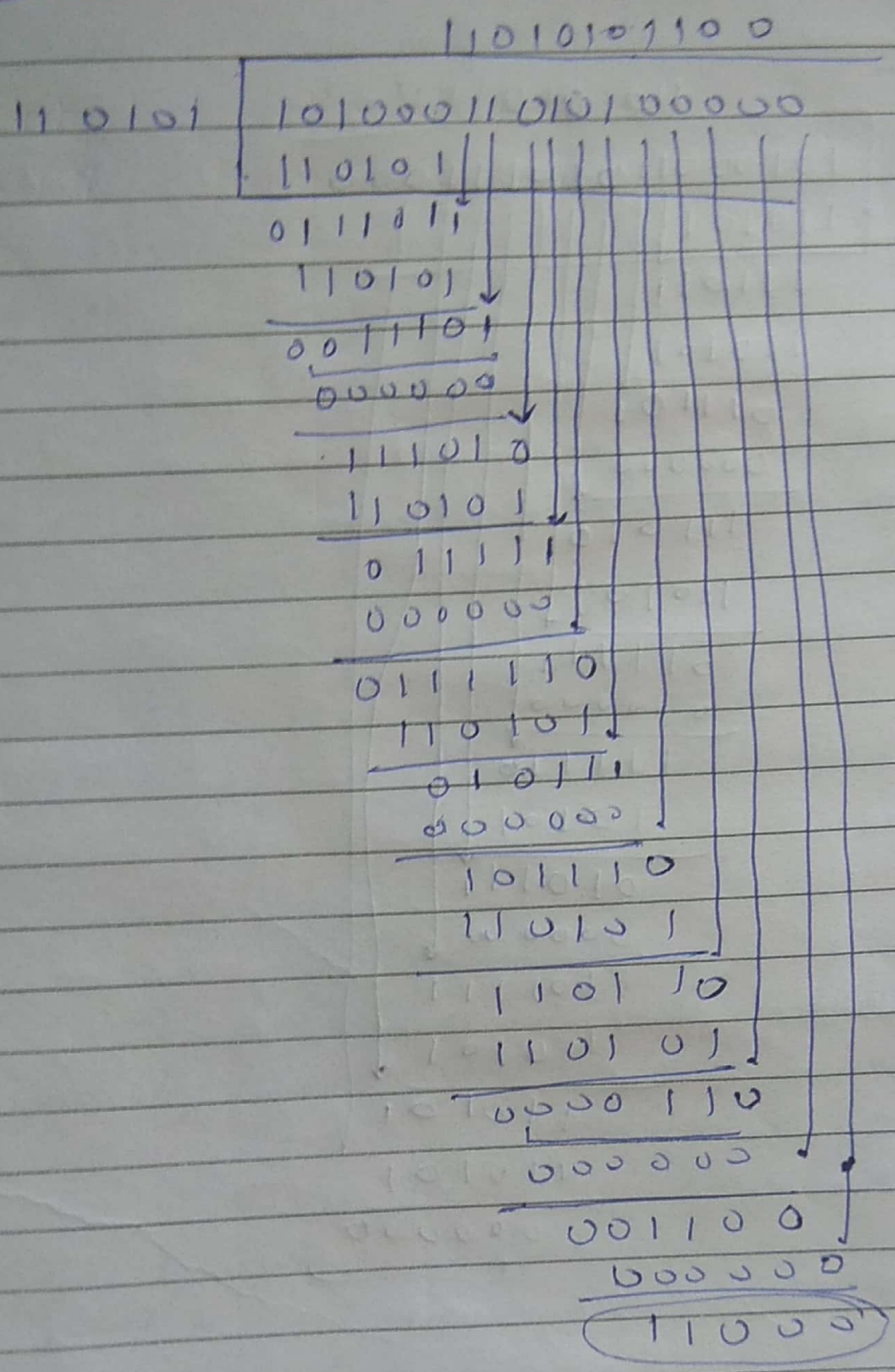
Two parts

↳ Local variable
↳ Data that can change

(•) Format of non-code part of executing subprogram is called activation record.

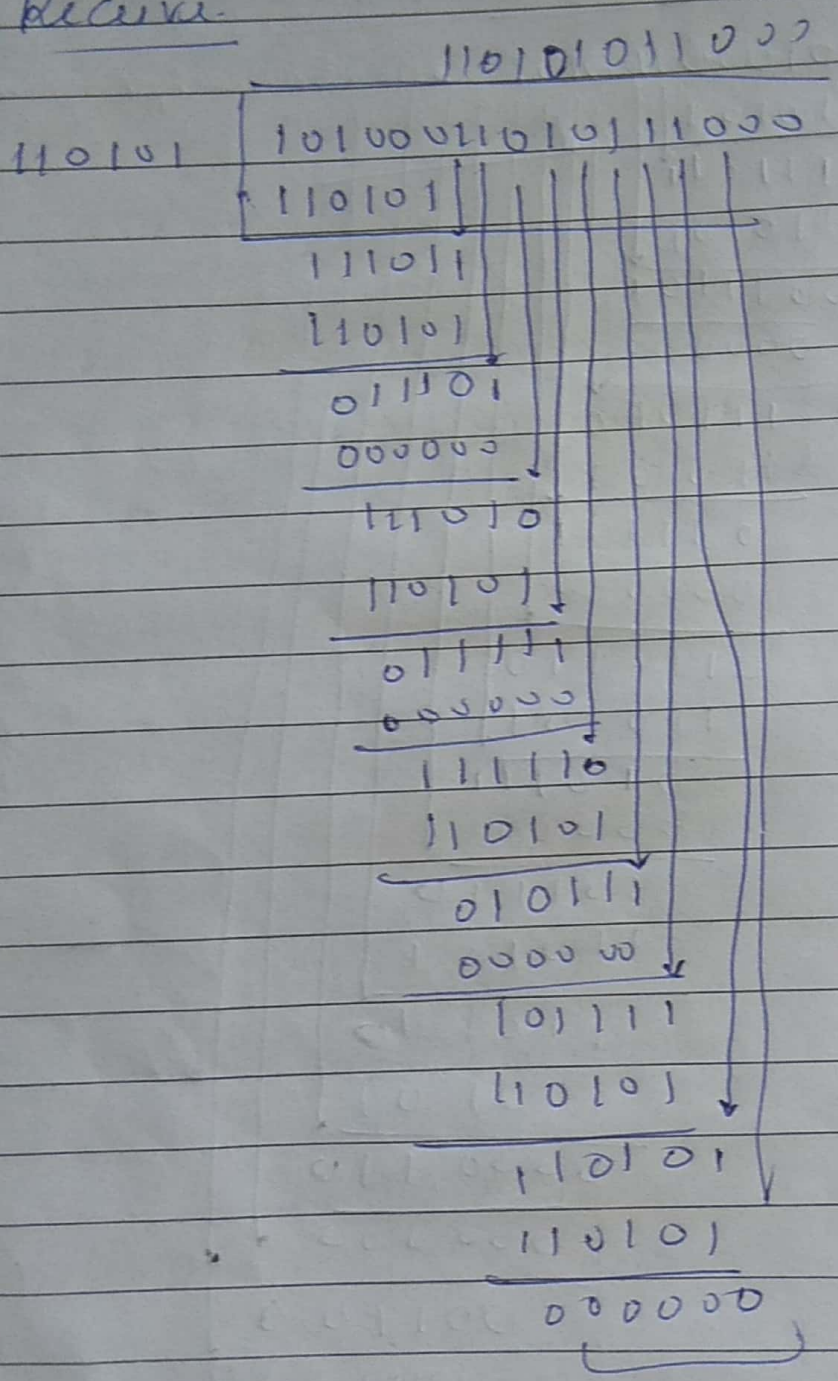
Date:
Page:

Synder



Date.
Page.

Receiver



1.0101

-) Databits 101000 | 10101

poly $x^5+x^4+x^2+1$

Data Control :-

Elements of data control :-

→ Referencing Environment — local / global
→ Visibility
→ Dynamic Scope — Static scope (local variable) / Dynamic scope (global variable)

*) Scope of variable x is the region of the program in which uses of x refers to its declaration.

*) Basic reason of scoping is to keep the variables in different parts of the program distributed from one another.

Dynamic scoping :-

It is not common in current programming languages partially, because it defeats - information hiding
- local analysis

* Global variable is a variable with a global scope meaning which is visible throughout the program.
* Set of all global variable is known as global environment or global state.

~~Static scoping~~ (logical

Advantage :- i) Writability
ii) More convenient & provides more flexibility than static scoping
iii) global variable exist only once in a script and visible in every function.

Static scoping (Lexical scoping) :-

i) Static variable always refers to its top level enviornments.

ii) It is also make much easier to make a modular code as programmer can figure out the scope by just by look at the code.

Q. What will the output of following pseudo code when parameters are passed by reference & dynamically scoped as assumed?

a=3; global

Ans- i)

```
void n(x)
{
  x=x*a;
  print x;
}

void m(y)
{
  a=1;
  a=y-a;
  n(a);
  print(a);
}

void main()
{
  m(a);
}
```

m(3)
↳ a=1
a=3-1=2
m(2)
↳ x=2*1

x 4
4 2

Storage Management :-

(i) Operations that requires storage allocation
ii) Component insertion & deletion opr
iii) Subprogram call & return operation
↳ a) creation or deletion of activation record
iv) explicit data structure creation & destruction operations.

Elements requiring storage

user defined data structure & constants
code segments for translated user program
system run time program (libraries).
input & output buffers
miscellaneous system data (eg:- tables, graphics)

Dynamic storage mgmt: heap

Date.
Page.

Static storage management

→ static allocation: (once allocated it does not change through execution)

Dynamic allocation

↳ heap storage management.

•) Heap storage

Memory used for dynamic allocation of data objects in unstructured manner is called heap storage.

Tasks of heap storage mngnt:

i) allocation
ii) recovery
iii) garbage collection
iv) compaction
v) ~~reuse~~ reuse

Fixed size & variable size elements.

Date.
Page.

Heap storage (Pros / Cons):-

Adv	Disadv
i) insertion easy	i) extra space for storing referencing counter
ii) Conceptually simple	ii) uses pointers to more time is required as every pointer assignment has to check count.
iii) immediate storage	iii) difficult to maintain.
	iv) decrease in execution efficiency

Stack / Heap

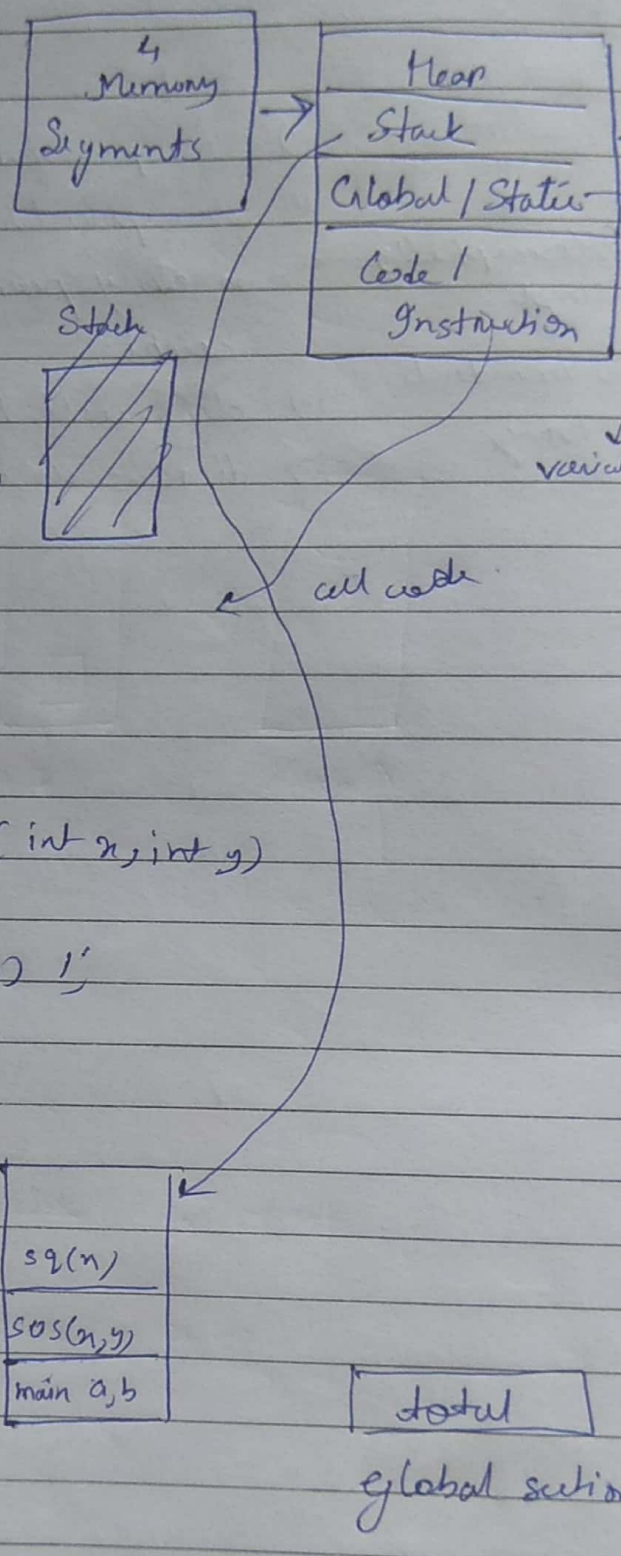
Date.
Page.

For ex

~~int square(int x)~~
~~{~~

```
int main()
{
   int a=4, b=8;
  total = sq of sum(a,b);
  cout / printf (o/p);
}

   int square (int x).
  { return x*x;
  }.
  int squares of sum( int x, int y)
    {
        in Z = sq(x+y) ;
        return z;
    }.
```



-) In stack ~~the~~ memory automatically reallocated
-) In heap user manually deallocate memory.

Date.
Page.

* Memory consists of four segments.

1) Dynamically allocated variables in heap.

→ contains information of function call.

Stack :- Stack is a stack of memory block.

i) They are created when function is called.
ii) So every time C++ program or any language call a function (including main) it allocates block of memory on the stack for this function.
iii) Block of memory contains every variable created on this function as well as the address of original function that called it.
iv) When execution of function ends it automatically deallocates this block of memory.

(V) For static memory allocation

Heap :-

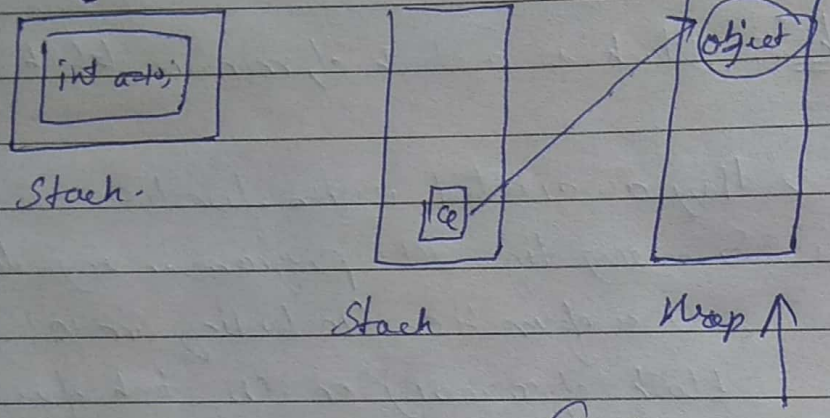
Whenever we created any object, it is always created in heap, and reference variable of object in stack.

Any object created in the heap has a global access & can be referenced from anywhere.

Test a = new Test;

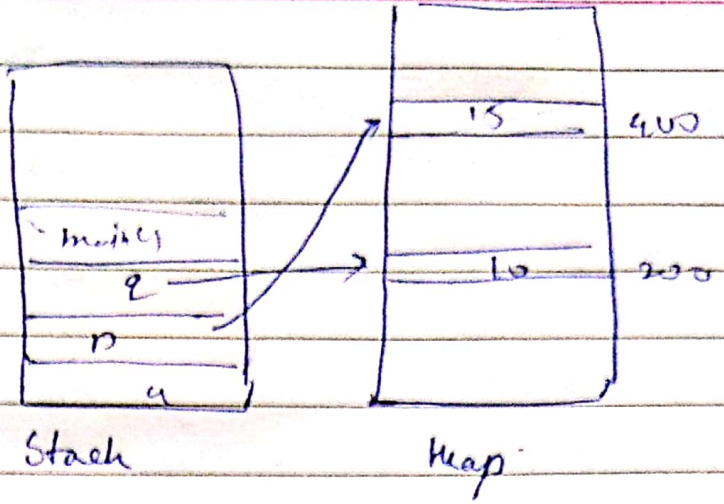
↳ reference variable for object.

int a=10; // local variable



~~Test a = new~~ Test a = new Test();

```
int main()
{ int a;
  int *p; *q
  p = (int *) malloc (size of, int);
  *p = 10;
  q = (int *) malloc (size of int);
  *q = 10;
  free(q);
  free (p);
}
```



Difference

Stack	Heap
i) It is used for static memory allocation.	i) It is used for dynamic memory allocation.
ii) Values on stack will be deleted automatically once function call ends.	ii) Values assigned in heap are stored permanently & should be deleted manually.
iii) Stack memory only contain local/primitive variables (Store actual values) and (reference variable) to object in heap space.	iii) reference/variables to objects are stored in
iv) Compile time.	iv) Run time.
v) Allocation of stack is much faster since all it really does move the stack pointer.	v) Heap mem. is slightly slower to be read from & written to because one has to use

1 3 4 5 -

Date.
Page.

Comparision in various general & special purpose programming languages:-

i) Fortran ii) C iii) Pascal iv) Smalltalk.

	C	Fortran	Pascal	Small talk.
Intended use	Application system, general purpose low-level op^v	Application, numerical computing	Application, Education	Application, general, busines, AI, Web
Objectoriented	No	Yes	No	Yes
Functional	No	No	No	Yes
Procedural	Yes	Yes	Yes	Yes
Standardised	1989, ANSI C89, ISO C90	1966, ANSI 66, ANSI 77, 15090 ISO 2003, 21010	1983 ISO	1998 ANSI
Invador	Dennis Ritchie	John Backus	Niklaus Wirth	Alan Kay, Dan Ingalls Adele Goldberg.
Platform independent	Yes	~~No~~ Yes	Yes	Yes.
		Invented originally to teach structured programming language with logical operations		Used in A.I. Disadv:- Libraries & ecosystem not good.

Fortran ~~Difference~~ disadvantage:-

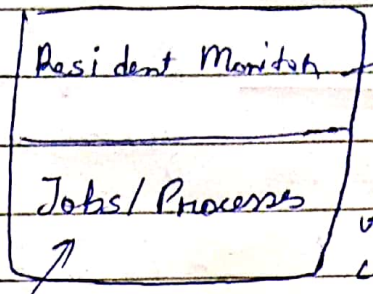
i) Poor string handling, concatenation is difficult
ii) Subroutine are passed by reference.
iii) Data scoping limited
iv) Loop control limited.

C Fortran Pascal Fortran

Batch Processing System

Date.
Page.

→ was used in recent times before



→ Set of instructions and that transfers the control from 1 job to another

when punch cards were processed or Input to a system

No. of jobs/tasks assigned to a system

*) Resident monitor includes more than one tasks when single processor is used to schedule the tasks automatically.

Batch operating system environment:

In this type of environment jobs with similar needs or requirements were batched or grouped together, and run through computer as a group, in this way it reduces the setup time ~~when~~ and CPU sit idle during job switching.

Small automatic job sequencing technique (resident monitor) was used which transfers the control from one job to another job automatically.

Advantage:-

i) Reduced set up time

Embedded system:

System:- Set of equipment, rules combine to form a system.

or

It is an arrangement in which all its units assemble & work together according to set of rules.

eg:- Watch is a display time system.

Embedded system

i) It is combination of software & hardware.
ii) It is designed to performed particular task.
iii) Task has to be completed in particular time.

eg:- Washing machine, mobile ~~machine~~, microwave.

Data flow language:-

In this methodology we have a stream of data, which is passed from instruction to instruction to be processed like procedure calls, condition execution system route the data to different instructions.

This could be seen as data flowing through, otherwise static instructions like how electric signals flow through circuits.

A data flow "if statement" could route the data to correct branch.

eg:- spreadsheets.
unix pipes

Pigs Latin

It is a procedural language or data flow language that helps you to process & transform the data.

Pig is a high level scripting language that is used in Apache-Hadoop. It enables data works to write complex data transformations without knowing Java.