C & C++ fundamentals

- * Variables and constants → Operands
  variables → identifiers
  constants → literals

- In switch case, the value of 'case' should be either int or string. floats are not allowed.

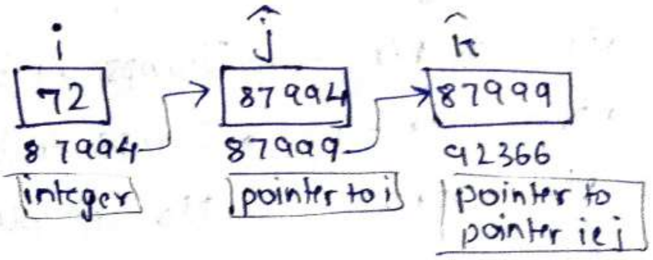
- Order of passing arguements in a fonction → Right to left.
  int a=1; printf("%d %d %d", a, a++, ++a);

$$\boxed{3 \quad 2 \quad 2}$$

← R to L

- * default return type of any fonction is int.
- * Pointers:

  int i = 72;
  int *j = &i;
  int **k = &j;



float *ptr; → This meant ptr is a pointer that stores address of a 'float' variable.

* → value at OR indirection operator.

- Storage Classes

Storage Class	Storage	Default value	Scope	Life
Automatic auto int;	RAM	Garbage Value	Block	Till control inside Block
Register register int;	CPU registers	Garbage Value	Block	Till control inside Block
Static static int	RAM	Zero	Block	Lifetime of program
External extern int;	RAM	Zero	Global	Lifetime of program.

default storage class → Automatic.

BY SHIVKARAMAN

- Macro Expression

#define PI 3.1428

PI ↓ Macro expression

3.1428 → Macro template

Macro expressions are replaced by their macro template ~~to~~ before compilation.

define AREA (x) (3.14 * x * x) → Macro expression with arguement.

- file incusion

include <file> ↔ Searched in only standard list of directories

include "file" → Standard list dir + Current dir.

- Pointer arithmatic

1) ptr = ptr +4 → Jumps 4 blocks ahead
2) ptr = ptr -4 → Jumps 4 blocks back
3) ptr1 - ptr2 → No of blocks b/w the two
4) Comparision of pointers → only == & != are allowed and only when both pointers belong to the same data type.

- Array initializations

int arr [5] = {1,2,3,4,5}

int arr[ ] = {1,2,3,4,5}

arr[2][2] = { {1,56}, {2,86} };

arr[2][2] = {1,56,2,86};

arr [ ][2] = {1,56,2,86};

In 2D arr, no of calumns is manditory

~~Pointers and arrays~~

* arr [i] ↔ *(arr+i) ↔ *(i+arr) ↔ i[arr]

* In 2D array → S[i][j]

S[i] → address of 0th element of ith row ie ith 1D array

*(S[i]+1) → 2nd element of ith row (value)

- Pointer ~~and~~ arrays

* arr[i] ↔ *(arr+i) ↔ *(i+arr) ↔ i[arr]

* 2D arrays → ~~arr~~ S[i][j]

Address of 1D array elements → S[i]
ie 0th element of each row

Value of 0th element of i'st row → *(S[i])

Address of 2nd 1D array elements i-e address of 1st element of ith row → • (S[i]+1)
⇕
• (*(s+i)+1)

∴ Address of jth element of ith row → (*(s+i)+j)
Value of jth element of ith row → ~~[illegible]~~ *(*(s+i)+j)

∴ $\boxed{S[i][j] \leftrightarrow *(*(s+i)+j)}$

- Pointer to 2D array → int (*p)[4]; → Pointer to 2D array of 4 columns
p= s;

- ~~Pointer of arrays~~ Array of pointers → int *p[4];

- Strings

·Initializations → char name[]= {'a', 'b', 'c', '\0'};
char name[] = "abc";
~~[illegible]~~ char *name = "abc";

- scanf("%[^\n]s", name); → input until it encounters '\n'

gets(), puts()

	name[4]="abc"; or name[ ] = "abc"	char*name="abc";
Reinitialization name="abcd";	Not allowed	Allowed
Modification name[1]='x';	Allowed	Not allowed

- String functions → strlen(str), strcpy(tar, src), strcat(tar, src);

  strcpy: tar = src,   strcat: → tar = tar + src;

  strcmp(str1, str2); str1 == str2 → -ve → s1<s2, 0 → s1=s2, +ve → s1>s2

- 2D string

  - str [i][j] → not optimal because lot of memory is wasted
  - Char * str [] → array of char pointers → Better approach.

- Sprintf and sscanf

sprintf → writes to a string	sscanf → reads from string

int i = 10;
float f = 3.14;
char ch = 'A';
char string[10];
sprintf(string, "%d %f %c", i, f, ch);
	char str[ ] = "shiva 18";
	int age;
	char name [10];
	sscanf(str, "%s %d", name, &age);

op: string → "$\underbrace{10}_{i}\underbrace{3.140000}_{f}\underbrace{A}_{ch}$"	op: name: "shiva" age: 18

- Unformatted i/o

getchar ()	putchar()	gets(string)
char ch = getchar()	putchar(ch)	puts(string)

fgets(base add of str, no of char including '\0', stdin)

- File I/O

  modes → read(r), write(w), append(a), read + write + modify (r+) →, write + modify (w+), read + append + cannot modify (a+)

  r, w r+, w+, a+ → FP points to beginning of file ie first character.

  a → point to last char.

Reading → character $\xrightarrow{ch=}$ fgetc (ptr) ↔ fscanf (ptr, "%c", ch)

writing → fputc (ch, ptr) ↔ fprintf (ptr, "%c", ch)

```
FILE * ptr = fopen ("file.ext", "mode");
```

- Pointer to functions:

```
syntax :  return-type  (* ptr) (args);
```

```
int add (int a, int b){
    return a+b;
}
```

```
int (*ptr) (int, int) = add;
 add (2,3);
 ptr (2,3);
```

if we remove paranthesis, it becomes a function decloration with return type int *

```
int * ptr (int, int);
```

- Dynamic memory allocation

1) malloc → int * ptr = (int *) malloc (n * sizeof (int));
→ returns base address (void pointer)

2) calloc → int * ptr = (int *) calloc (n, sizeof (int));
→ malloc + initialises all elements to zero

3) free → free (ptr) → frees memory allocated

4) realloc → int * ptr = (int*) realloc (ptr, newsize * sizeof (int));

ID array

```
int *arr = (int*) malloc (n* sizeof(int));
arr[0]=1;
*(arr+ 2)= 10;
```

2D array

```
int **array = (int **) malloc (rows *
                      sizeof (int *));
for (int i=0;  i<rows; i++){
   arr[i] = (int *) malloc (col*
                      sizeof (int);
}
```

Pointers

Wild pointer

* Not initialized while declaring
* int * ptr;
* stores garbage val

NULL

* int * ptr = NULL;
int * ptr = 0;

void pointer

* Can point to any data-type variable.
* void * ptr;
* Can be typecasted to desired datatype

Dangling pointers

* Points to a deleted/ freed memory

- <u>Datatypes and size</u>

int, unsigned int, long int, unsigned long int, float → 4B
short int, unsigned short int → 2B
char, unsigned char → 1B
double, long long int, unsigned long long int → 8B
long double → 10B

Range → n bits : signed : -2^{n-1} to $2^{n-1}-1$
unsigned : 0 to 2^{n}

- <u>Operators</u> : Arithmatic, comparision, logical, assignment, bitwise

- <u>Scope resolution operator (::)</u> → global scope cout << :: x;

- <u>Reference variable (&)</u> → Giving another name. Acts like internal pointer
→ Should be initialised during declaration.

- <u>Inline functions</u> → Used when execution time < function switching time
→ $\boxed{\text{inline int sum(int a, int b) \{\}}}$;

- <u>Iterators</u>

$\boxed{\text{vector <int> :: iterator it;}}$ → Syntax

it++; → Points to next iterator

it = it + 1 → Points to Next memory location.

- Time complexity and constraints

Max iterations <= 10^7

- Range of data type

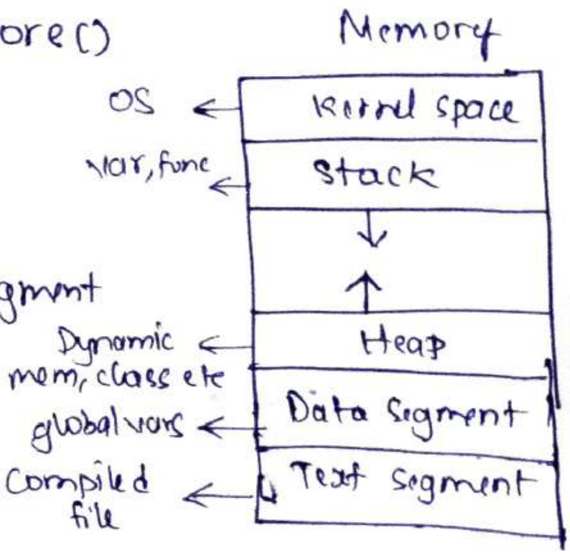
int → 10^{-9} to 10^{+9}

long int → 10^{-12} to 10^{+12}

long long int → 10^{-18} to 10^{+18}

- In c++, gets() → getline(cin, str);

fflush(stdin) → cin.ignore()



- Array size limit

locally → 10^5 → Stack

globally → 10^7 → Data Segment

- STL data structures

1) Vector: push_back()
pop_back()
size() } O(1)

.erase(val)
.clear() } O(n)

2) map:

map<key, value> → Implimented using RED BLACK TREES

mp.insert({k,v});
mp.erase (key);
mp.find (key); } O(logn)

mp.empty(), mp.size() → O(1)

mp.clear() → O(n)

unordered_map<key, value> → Implimented using hash table

insert, erase, find → O(1)

- Important questions

digit that can be used in a system is 0 − (base −1)
in binary → base = 2 $(x)_2$
so digits → 0−(2-1) i.e 0-1
in decimal → base = 10, digits → 0-9
in octal → base = 8, digits 0-7
in hexadecimal → base = 16, digits 0-15

- Conversions
1) Binary to decimal
Decimal to binary
2) Decimal to octal
Octal to decimal
3) Decimal to hexadecimal
Hexa decimal to decimal

1) Binary to decimal; → Multiply by 2^x, → x → element index

$(1011)_2 \rightarrow 1\times2^0 + 1\times2^1 + 0\times2^2 + 1\times2^1$
(3 2 1 0)
$= (11)_{10}$

Decimal to binary → 25 = 11001
16 8 4 2 1

2 | 25
2 | 12 − 1
2 | 6 − 0
2 | 3 − 0
  | 1 − 1

~~0.825 × 8 = 6.600~~

2) Decimal to octal →
1) $(658.825)_{10} \rightarrow (\ )_8$

8 | 658
8 | 82 − 2
8 | 10 − 2
8 | 1 − 2
  | 0 − 1

$(1222.64631)_{10}$

	carry
0.825 × 8 = 6.600	6
0.6 × 8 = 4.8	4
0.8 × 8 = 6.4	6
0.4 × 8 = 3.2	3
0.2 × 8 = 1.6	1

In octal system, a number contains only digits from $\boxed{0-7}$

BY SHIVKARAMAN

DATE: PAGE:

Octal to decimal

$\underset{8^3 8^2 8^1 8^0}{1222} \cdot \underset{8^{-1} 8^{-2} 8^{-3} 8^{-4}}{6463}$

$1222 = 2 \times 8^0 + 8^1 \times 2 + 8^2 \times 2 + 8^3 \times 1$

$= 2 + 16 + 128 + 512$

$= 658$

$\underset{8^{-1} 8^{-2} 8^{-3} 8^{-4}}{0.6463} = 8^{-1} \times 6 + 8^{-2} \times 4 + 8^{-3} \times 6 \times 8^{-4} \times 3$

64×8
512

8) Decimal to hexa-decimal

$(5386.345)_{10} - (?)_{16}$

16 | 5386
16 | 336 — $\boxed{10}$ — A
16 | 21 — 0
16 | 1 — 5
   | 0 — 1

$(\boxed{150A.585})_{16}$

$0.345 \times 16 = 5.52 - \boxed{5}$

$0.52 \times 16 = 8.32 - \boxed{8}$

$0.32 \times 16 = 5.12 - \boxed{5}$

Hexa decimal to decimal

$\overset{2\,1\,0}{1A2} \cdot \overset{-1\,-2}{10}$

$1A2 = 2 \times 16^0 + 10 \times 16^1 + 1 \times 16^2$

$= 418$

$0.10 = 16^{-1} \times 1 + 0$

$= 0.065$

$(1A2.10)_{16} = (418.065)$

BY SHIVKARAMAN