Programming with C

Welcome to the Workshop

Programming with C

Objective

Gain practical coding experience with C

Fundamental of software development

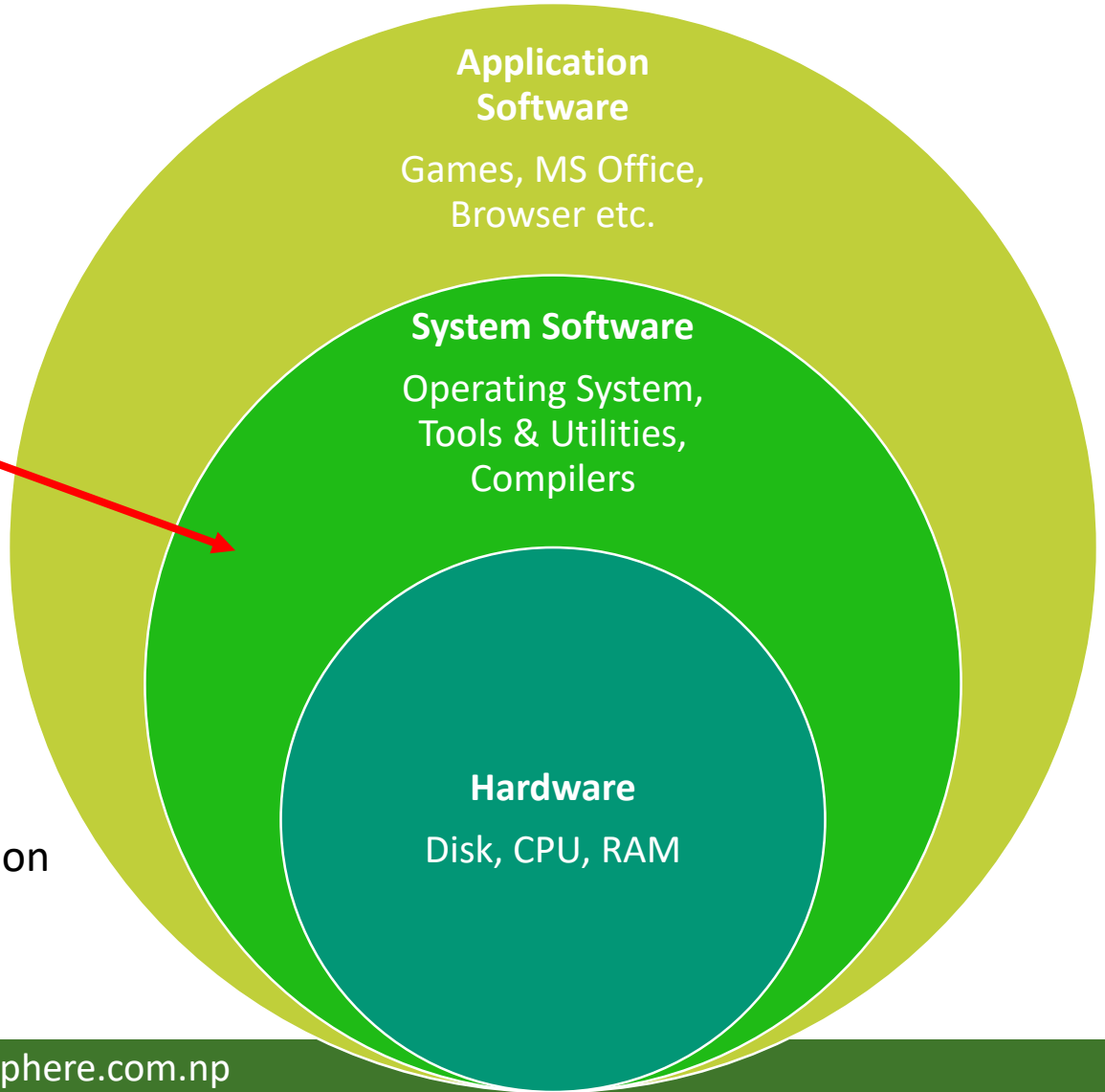
C

System Software

https://whatis.techtarget.com/definition/system-software

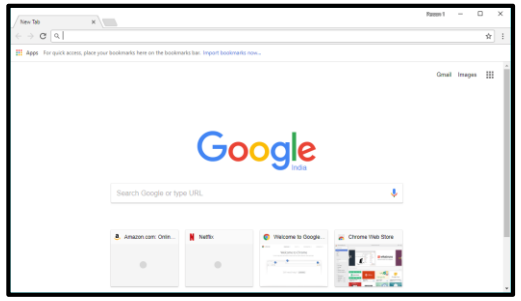
Application Software

https://searchsoftwarequality.techtarget.com/definition/application



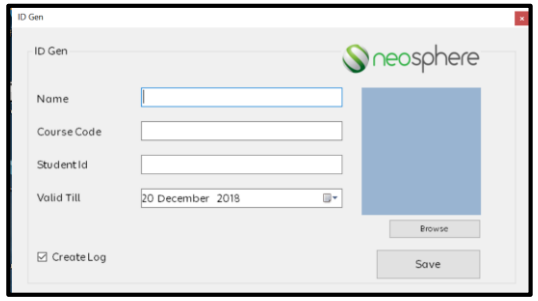
Types of **Application**

Languages



Web

- Java
- .NET
- PHP
- Python
- Others



Desktop

- Java
- .NET
- Python
- Others

Mobile

- Android (java, Kotlin)
- iOS (Swift)

MySQL®

Office Access

Microsoft® SQL Server®

SQLite

ORACLE® DATABASE

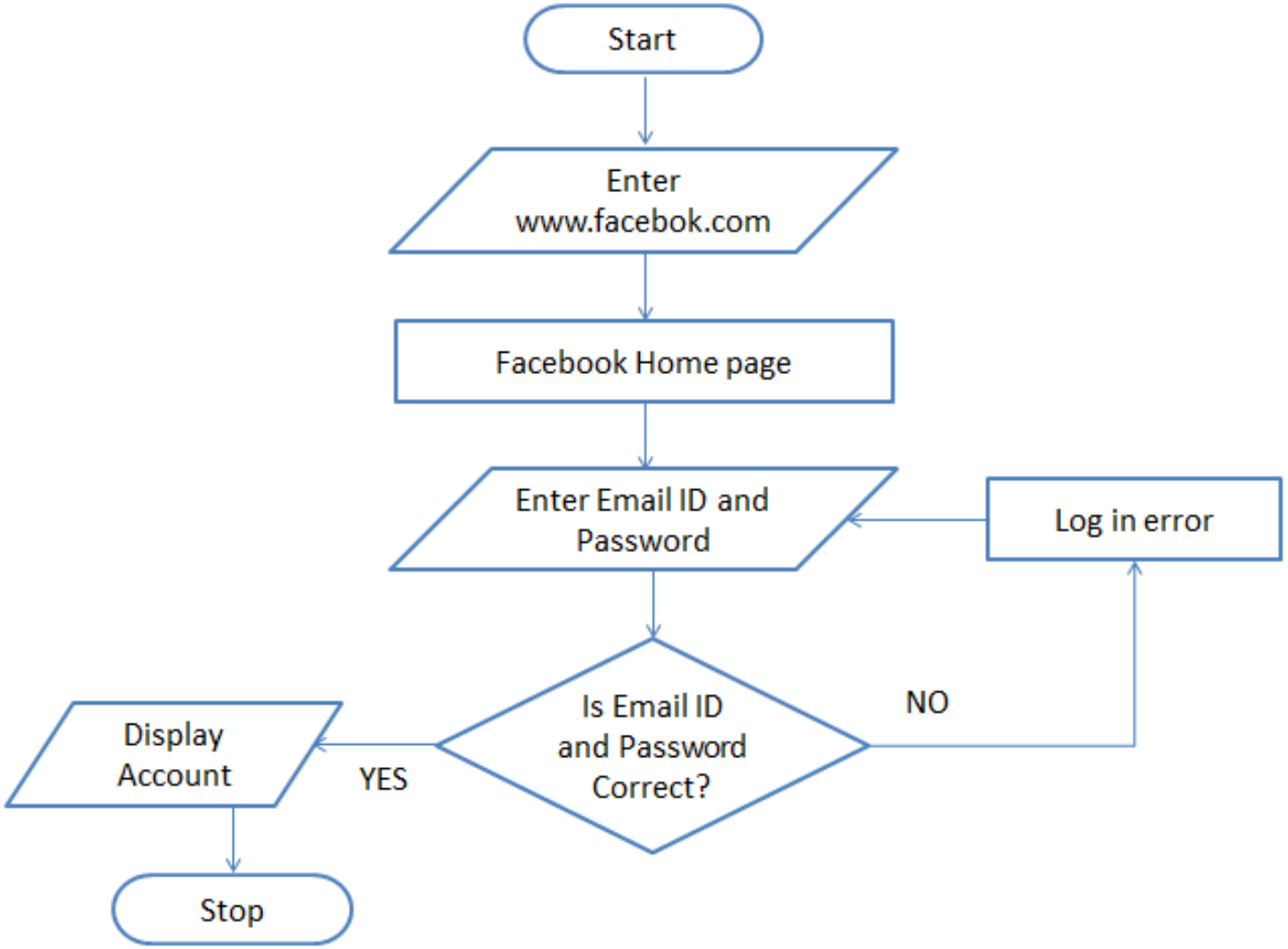
Firebase Realtime Database

Algorithm

- How to solve a class of problems.

- Algorithms can perform calculation, data processing, and automated reasoning tasks

Facebook Login Algorithm



Question?

Algorithms can perform calculation, data processing, and automated reasoning tasks

Symbols and Description

Symbol	Name	Function
	Start/end	An oval represents a start or end point
	Arrows	A line is a connector that shows relationships between the representative shapes
	Input/Output	A parallelogram represents input or output
	Process	A rectangle represents a process
	Decision	A diamond indicates a decision

Tools to generate flowchart

https://www.draw.io/

Q: Compare Two Numbers

Introduction to C Programming

- C is a procedural programming language
- Developed by Dennis Ritchie
- Mainly developed as a system programming language to write operating system

preprocessor directive

Header File

```
#include <stdio.h>
void main()
{

printf("welcome to C Programming");

}
```

return type

function

main function block

\n is used to move the control onto the next line

\t is used to give a horizontal tab i.e. continuous five spaces

#include Directive

#include directive tells the preprocessor to insert the contents of another file into the source code at the point where the **#include** directive is found

Question

Question:

Write a Program to print your name and address

Escape Characters

Symbol	Description
\n	New Line
\t	Horizontal tab
\v	Vertical Tab
\a	Alarm
\\	\
\b	Backspace

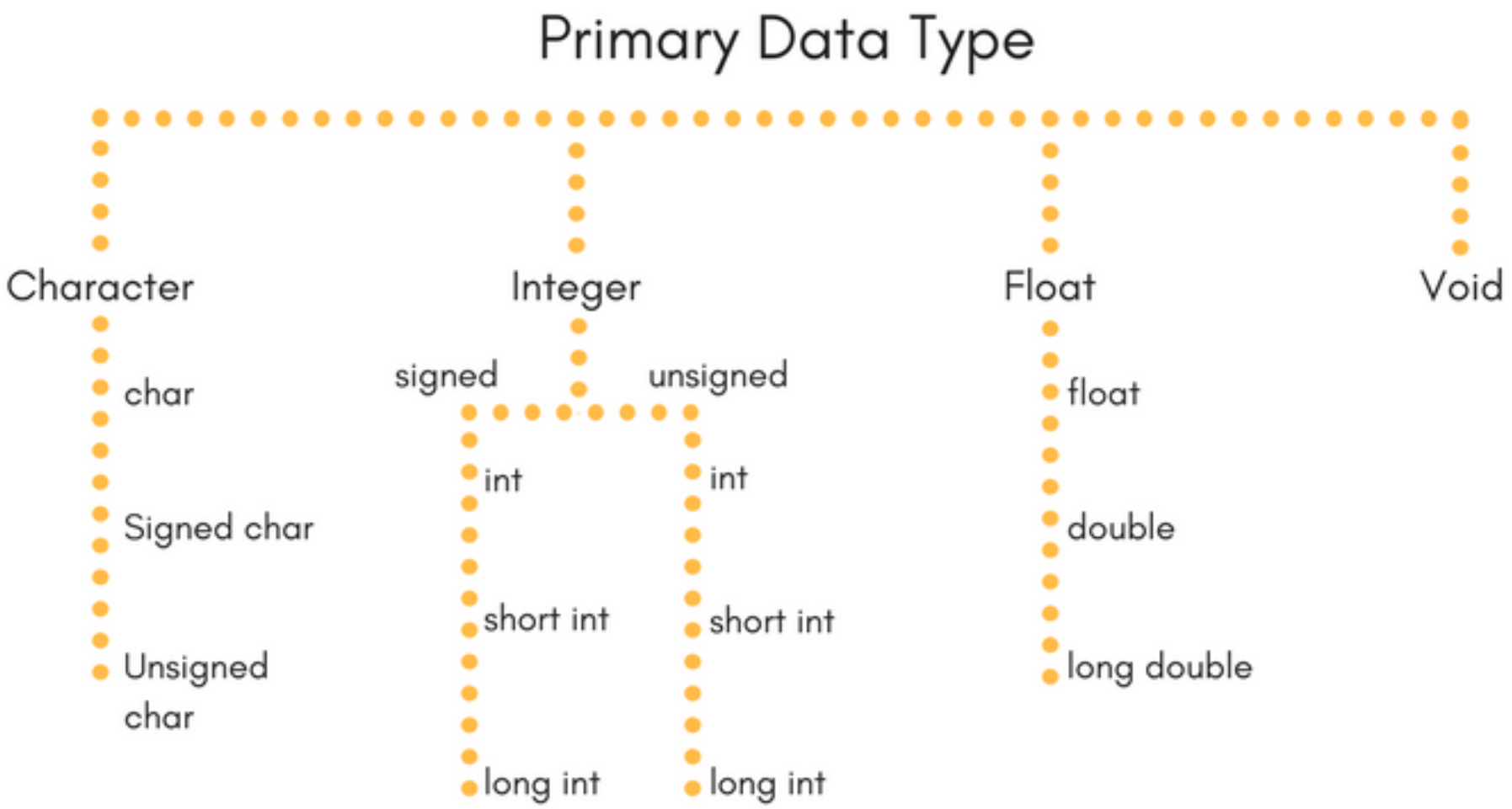
Data Types

Primary data types:

fundamental data types in C namely integer

Derived data types:
array, structure, union and pointer

Data Types


Primary Data Type
Character
Integer
Float
Void
char
Signed char
Unsigned char
signed
unsigned
int
short int
long int
int
short int
long int
float
double
long double

Integer

Type	Size(bytes)	Range
int or signed int	2	-32,768 to 32767
unsigned int	2	0 to 65535
short int or signed short int	1	-128 to 127
unsigned short int	1	0 to 255
long int or signed long int	4	-2,147,483,648 to 2,147,483,647
unsigned long int	4	0 to 4,294,967,295

Floating Point Type

Type	Size(bytes)	Range
Float	4	3.4E-38 to 3.4E+38
double	8	1.7E-308 to 1.7E+308
long double	10	3.4E-4932 to 1.1E+4932

Character type

Type	Size(bytes)	Range
char	1	-128 to 127

Variables

Assignment Operator

int a = 10;

Data Type

Variable

Value

Format Specifier

Format specifier	Description	Supported data types
%c	Character	char unsigned char
%d	Signed Integer	short unsigned short int long
%e or %E	Scientific notation of float values	float double
%f	Floating point	float

Operators

An operator is a symbol which operates on a value or a variable. For example: + is an operator to perform addition.

Operators

1. Arithmetic Operators
2. Increment and Decrement Operators
3. Assignment Operators
4. Relational Operators
5. Logical Operators
6. Bitwise Operators
7. sizeof Operator
8. Ternary Operator

Program

```
void main()
{
   int a = 9,b = 4,c;
    c = a+b;
    printf("a+b = %d \n",c);
}
```

Break
15 mins

Input from user

```
void main()
{
    int a,b;
    printf("Input any number");
    scanf("%d%d",&a,&b);
    printf("Number is %d %d",a,b);
}
```

a → 10 &a					
					30 &b

b

Assignment Operator

Operator	Example	Same as
=	a = b	a = b
+=	a += b	a = a+b
-=	a -= b	a = a-b
*=	a *= b	a = a*b
/=	a /= b	a = a/b
%=	a %= b	a = a%b

Relational Operator

Operator	Meaning of Operator	Example
==	Equal to	5 == 3 returns 0
>	Greater than	5 > 3 returns 1
<	Less than	5 < 3 returns 0
!=	Not equal to	5 != 3 returns 1
>=	Greater than or equal to	5 >= 3 returns 1
<=	Less than or equal to	5 <= 3 return 0

Logical Operator

Operator	Meaning of Operator	Example
&&	Logial AND. True only if all operands are true	If c = 5 and d = 2 then, expression ((c == 5) && (d > 5)) equals to 0.
||	Logical OR. True only if either one operand is true	If c = 5 and d = 2 then, expression ((c == 5) || (d > 5)) equals to 1.
!	Logical NOT. True only if the operand is 0	If c = 5 then, expression ! (c == 5) equals to 0.

Bitwise

Operators	Meaning of operators
&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR
~	Bitwise complement
<<	Shift left
>>	Shift right

Ternary Operator (?:)

conditionalExpression ? expression1 : expression2

Question

Write a program to add two numbers

Control Statement

- Decision making statements
- Selection statements
- Iteration statements
- Jump statements

Control Statement

- Decision making statements
- Selection statements
- Iteration statements
- Jump statements

If Statement

```
if (condition)
{
  statements
}
```

If…else Statement

```
if (condition)
{
  statements;
}else{
Statement/s;
}
```

Question

Write a program to compare two numbers…

if the number is even or Odd

If…else if…else Statement

```
if(condition1)
{
// statement(s);
}
else if(condition2)
{
//statement(s);
}
……
else
{
//statement(s);
}
```

Switch

```
switch( expression )
  {
        case constant-expression1:
        statements1;
        break;
        case constant-expression2:
        statements2;
        break;
        case constant-expression3:
        statements3;
        break;
        default :
        statements4;
  }
```

Loops

Entry Controlled	Exit Controlled
▪ while ▪ for	▪ do…while

While

```
int n, i=1;
clrscr ();
printf ("enter the values of n");
scanf ("%d", &n);
printf ("natural numbers from 1 to %d ", n);
while (i<=n)
{
        printf ("%d\ t", i);
        i++;
}
```

for

```
int n,i;
        clrscr();
        printf("enter any number");
        scanf("%d",&n);
        for(i=1,i<=n,i++)
        {
                printf(“%d”,i);
        }
```

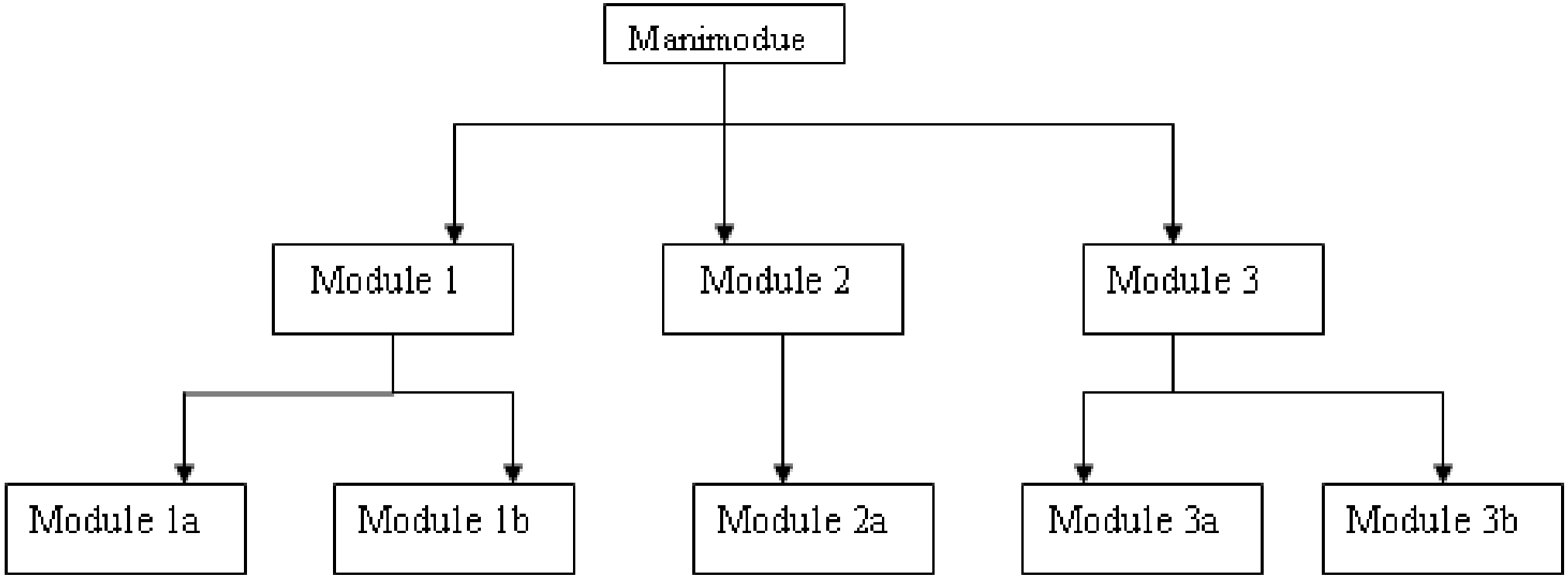
Break
Day-1

do…while

```
int n,i=1;
        clrscr();
        printf("enter any number");
        scanf("%d",&n);
        printf("natural numbers from 1 to %d",n);
        do
        {
                printf("%d\t",i);
                i++;
        }while(i<=n);
```

Print multiplication table of any numbers from 1 to 20

Function


Manimodue
Module 1
Module 2
Module 3
Module 1a
Module 1b
Module 2a
Module 3a
Module 3b

Function

We have 2 types of functions:

1. Standard library functions
2. User-defined functions

Function

- Function declaration
- Function calling
- Function definition

Function

```
return type function_name(argument list)
{
code block;
}
```

Function

```
#include < stdio.h>
#include < conio.h>
void speakHello();      /* Function Declaration */
void main()
{

  clrscr();
 speakHello();      /* Function Calling */
  getch();
}

void speakHello()      /* Function Definition*/
{
  printf("\n WELCOME TO THE WORLD OF C LANGUAGE");
}
```

Function –return statement

```
#include < stdio.h>
void speakHello(int, int);      /* Function Declaration */
void main()
{

   clrscr();

printf("%d", speakHello(12,21));      /* Function Calling */
   getch();
}

void speakHello(int x, int y)      /* Function Definition*/
{
return (x+y);
}
```

Array

```
type variable_name[lengthofarray];

int min[9];
```

Array

```
int min[5];
```

45	56	89	11	22
min[0]	min[1]	min[2]	min[3]	min[4]

CONTINOUS MEMORY LOCATIONS

Array

```
int myArray[5] = {1,2,3,4,5};

for (int i=0;i<5;i++){
   printf("%d", myArray[i]);
}
```

Question

Find the greatest number in an array

Array – Multidimensional

int values [3] [4]

[0,0]	[0,1]	[0,2]	[0,3]	[0,4]
[1,0]	[1,1]	[1,2]	[1,3]	[1,4]
[2,0]	[2,1]	[2,2]	[2,3]	[2,4]

Size: 3x4 = 12

Array – Multidimensional

```
int values [3] [4] = {
        {
                1, 2, 3, 4
        }
        {
                5, 6, 7, 8
        }
        {
                9, 10, 11, 12
        }
};
```

Size: 3x4 = 12

Find the sum of matrix

```
for(i=0;i<r;++i)
    for(j=0;j<c;++j)
    {
        sum[i][j]=a[i][j]+b[i][j];
    }
```

Pointers

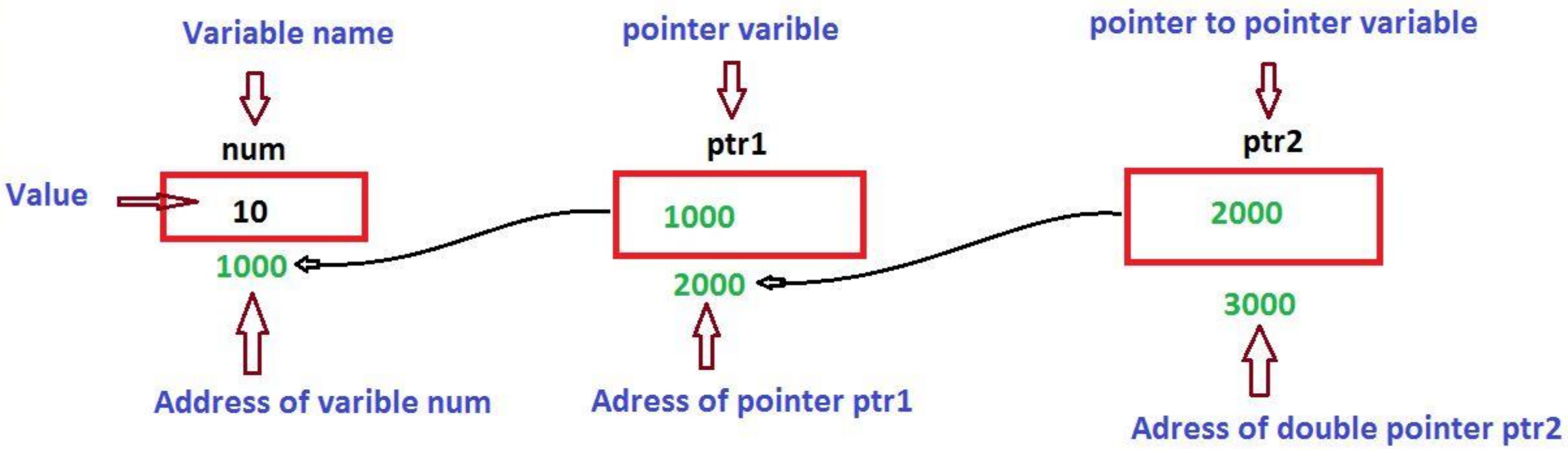
pointer is a variable that contains an address which is a location of another variable in memory

Pointers

Datatype *pointer name

```
int *p
```

The asterisk (*) tells the variable ‘p’ is a pointer variable.
‘p’ needs a memory location.
‘p’ points to a variable of type datatype.


Variable name
pointer varible
pointer to pointer variable
num
ptr1
ptr2
Value
10
1000
2000
1000
2000
3000
Address of varible num
Adress of pointer ptr1
Adress of double pointer ptr2

Pointer

```
int a=10;
int *b;
        b=&a;
        printf("value of a=%d \n",a);
        printf("value of a=%d \n",*(&a));

        printf("value of a=%d \n",*b);
        printf("address of a=%u \n",&a);
        printf("address of a=%u \n",b);
        printf("address of b=%u \n",&b);
        printf("value of b=address of a=%u \n",b);
```

Pointer- Arithmetic

```
int main()
{
    int m = 5, n = 10, o = 0;
    int *p1;
    int *p2;
    int *p3;

    p1 = &m;    //printing the address of m
    p2 = &n;    //printing the address of n
    printf("p1 = %d\n", p1);
    printf("p2 = %d\n", p2);

    o = *p1+*p2;
    printf("*p1+*p2 = %d\n", o);//point 1
```

Pointer- Arithmetic

```
p3 = p1-p2;
   printf("p1 - p2 = %d\n", p3); //point 2

   p1++;
   printf("p1++ = %d\n", p1); //point 3

   p2--;
   printf("p2-- = %d\n", p2); //point 4

   //Below line will give ERROR
   printf("p1+p2 = %d\n", p1+p2); //point 5

   return 0;
}
```

File Handling in C

- Creation of a new file (fopen with attributes as “a” or “a+” or “w” or “w++”)
- Opening an existing file (fopen)
- Reading from file (fscanf or fgetc)
- Writing to a file (filePointerrintf or filePointeruts)
- Moving to a specific location in a file (fseek, rewind)
- Closing a file (fclose)

Code

https://github.com/ratneshkr/cpro/blob/master/filehandle.c

Download Workshop Content

https://github.com/ratneshkr/cpro

Thank You