KALIYONA

Programming in C

Introduction

Explained in Kannada

ಕನ್ನಡದಲ್ಲಿ

Watch Now

Introduction to C

- C is a very popular programming language
- Created by Dennis Ritchie in 1972
- If you know C, you can learn any other programming language
- C is very fast compared to other programming languages
- C is very versatile

Setup & Installation

- Editor
  - Visual Code
  - Code::Blocks
- Compiler
  - GCC
- Online Compiler
  - Ex: Tutorialspoint, onlinegdb, programiz etc

Steps to write your first program

- Add headers

```
#include <stdio.h>
```

- Write main( ) function

```
int main( ) {
}
```

- Add printf( ) statement

```
printf( )
```

Steps to write your first program

- Add message

```
printf("Hello World");
```

- Add return statement

```
return 0;
```

- Compile and Run

First Program

```
#include <stdio.h>


int main( ) {
      printf("Hello World");
      return 0;
}
```

KALIYONA

Programming in C

Variables & Datatypes

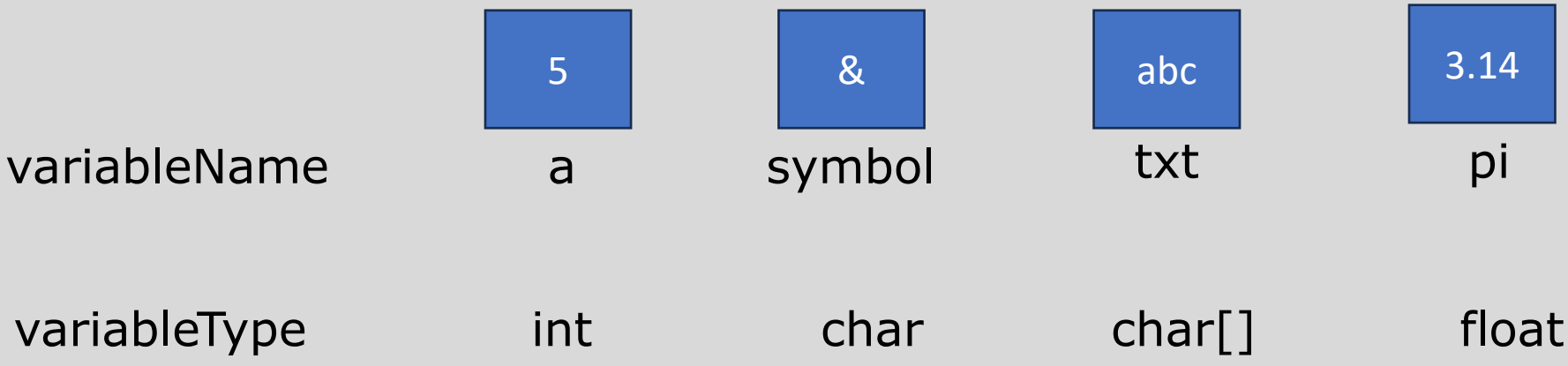
Explained in Kannada

ಕನ್ನಡದಲ್ಲಿ

Watch Now

Variables

- Containers that store some information




```
int a = 5;
char symbol = '&';
```

Variables - Rules

- Variables should have a name and a defined type
- Variable names are case sensitive
- First character should always be an alphabet or '_'
- No other character than '_' is allowed
- No comma or blank space is allowed
- Names must be meaningful

Datatypes

DataType	Size
int	2 or 4 bytes
float	4 bytes
double	8 bytes
char	1 byte

Constants

- Are variables whose value does not change
- ‘const’ keyword is used to define constants

```
const int age = 22;
```

Keywords

- Pre-defined words
- 32 Keywords

auto	break	case	char	const	continue	default	do
double	else	enum	extern	float	for	goto	if
int	long	register	return	short	signed	sizeof	static
struct	switch	typedef	union	unsigned	void	volatile	while

Comments

- Are used to add some additional information about the code
- Single Line Comments

```
// This is a single line comment
```

- Multi Line Comments

```
/*
    This
    is a
    Multi line
    Comment
*/
```

Points to Remember

- C is a case-sensitive programming language
- Each line ends with ‘;’
- Program is executed line by line
- main( ) function is the starting point of execution

KALIYONA

Programming in C

Input/Output

Explained in Kannada

ಕನ್ನಡದಲ್ಲಿ

Watch Now

Output

- ‘printf()’ function is used to display output messages
- “\n” is used to add new line

```
printf(“Hello World”);
printf(“Welcome to Kaliyona\n”);
```

Format Specifiers

DataType	Format Specifier
int	%d or %i
float	%f
double	%lf
char	%c
String	%s

Input

- “scanf( )” function is used to get the user input
- Takes 2 arguments
    - Format specifier
    - Reference operator
- It can take multiple inputs too.

```
scanf(“%d”, &age);
```

Code:

```
#include <stdio.h>

int main() {
    int num = 30;
    float pi = 3.14;
    int age, num1, num2;

    printf("Hello World\n");
    printf("Welcome to Kaliyona\n");
```

contd..

Code:

```
    printf("Number value is %d", num); //Format Specifier
    printf("\n PI Value is %f", pi);

    printf("\nEnter the age of the user ");
    scanf("%d", &age);
    printf("The age of the user is %d", age);

    printf("\nEnter two numbers ");
    scanf("%d %d", &num1, &num2);

    return 0;
}
```

KALIYONA

Programming in C

Operators

Explained in Kannada

ಕನ್ನಡದಲ್ಲಿ

Watch Now

Arithmetic Operators

Operator	Name	Description
+	Addition	Adds two numbers
-	Subtraction	Subtracts one number from another
*	Multiplication	Multiplies two numbers
/	Division	Divides one number by another
%	Modulus	Returns the Remainder value
++	Increment	Increments the number by 1
--	Decrement	Decrements the number by 1

Comparison Operators

Operator	Name	Description
==	Equal to	Returns true if two values are equal
!=	Not equal	Returns true if two values are not equal
>	Greater than	Returns true if the first value is greater than to the second
<	Less than	Returns true if the first value is less than to the second
>=	Greater than or equal to	Returns true if the first value is greater than or equal to the second
<=	Less than or equal to	Returns true if the first value is less than or equal to the second

Logical Operators

Operator	Name	Description
&&	Logical And	Returns true if both statements are true
||	Logical Or	Returns true if any one statement is true
!	Logical Not	Reverses the result. Returns 1 if the result is 0

Code:

```
#include <stdio.h>

int main()
{
    int num1= 10;
    int num2 = 5;

    int res;

    //Arithemetic Operators
    // res = num1 + num2;
    // printf("Addition result is %d", res);

    printf("Addition result is %d\n", num1 + num2);
    printf("Subtraction result is %d\n", num1 - num2);
    printf("Multiplication result is %d\n", num1 * num2);
    printf("Division result is %d\n", num1 / num2);
    printf("Remainder value is %d\n", num1 % num2);
    printf("Incrementing num1 %d\n", ++num1);
    printf("Decrementing num2 %d\n", --num2);
```

contd..

Code:

```
//Comparison Operators
printf("Are numbers equal %d\n", num1 == num2);
printf("Are numbers not equal %d\n", num1 != num2);
printf("num1 is greater than or equal to num2 %d\n", num1 >= num2);
printf("num1 is lesser than or equal to num2 %d\n", num1 <= num2);

//Logical Operators
/*
Logical AND
true - false = false
false - true = false
false - false = false
true - true = true
*/

res = (num1 > 0) && (num1 > num2);
printf("AND result is %d\n", res);
```

contd..

Code:

```
/*
    Logical OR
    false - false = false
    true - false = true
    false - true = true
    true - true = true
    */
    res = (num1 < 0) || (num1 < num2);
    printf("OR result is %d\n", res);

    /*
    Logical not
    true = false
    false = true
    */
    res = !(num1 > num2);
    printf("Logical NOT result is %d", res);

    return 0;
}
```

KALIYONA

Programming in C

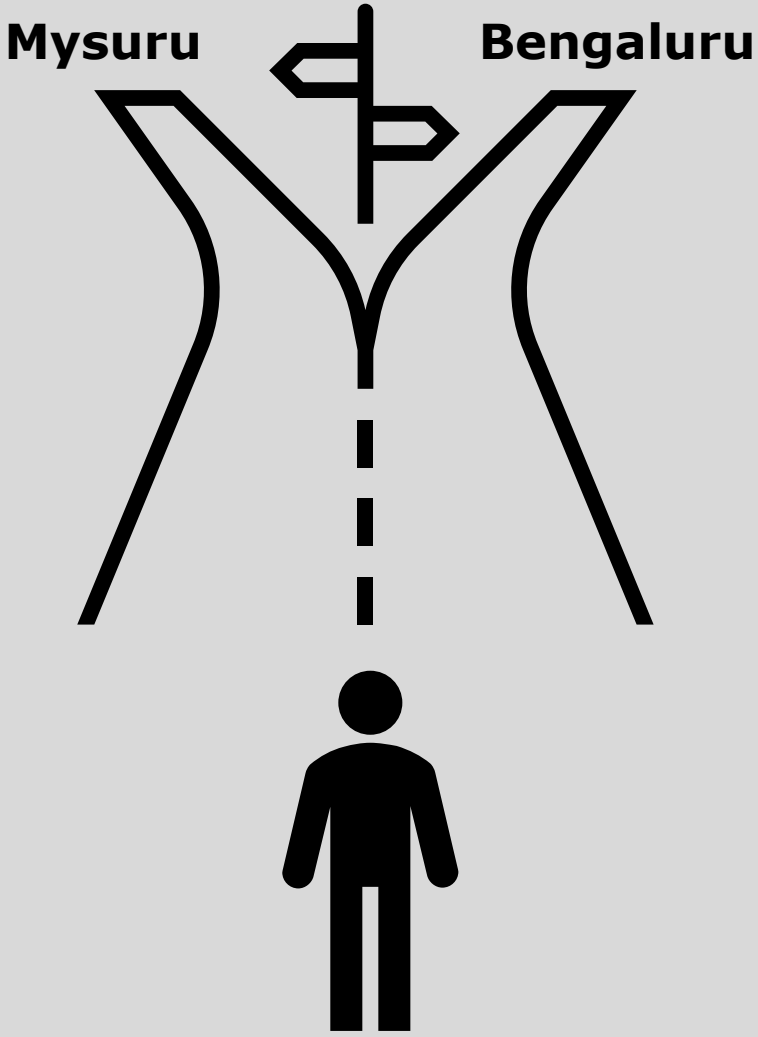
Conditional Statements

Explained in Kannada

ಕನ್ನಡದಲ್ಲಿ

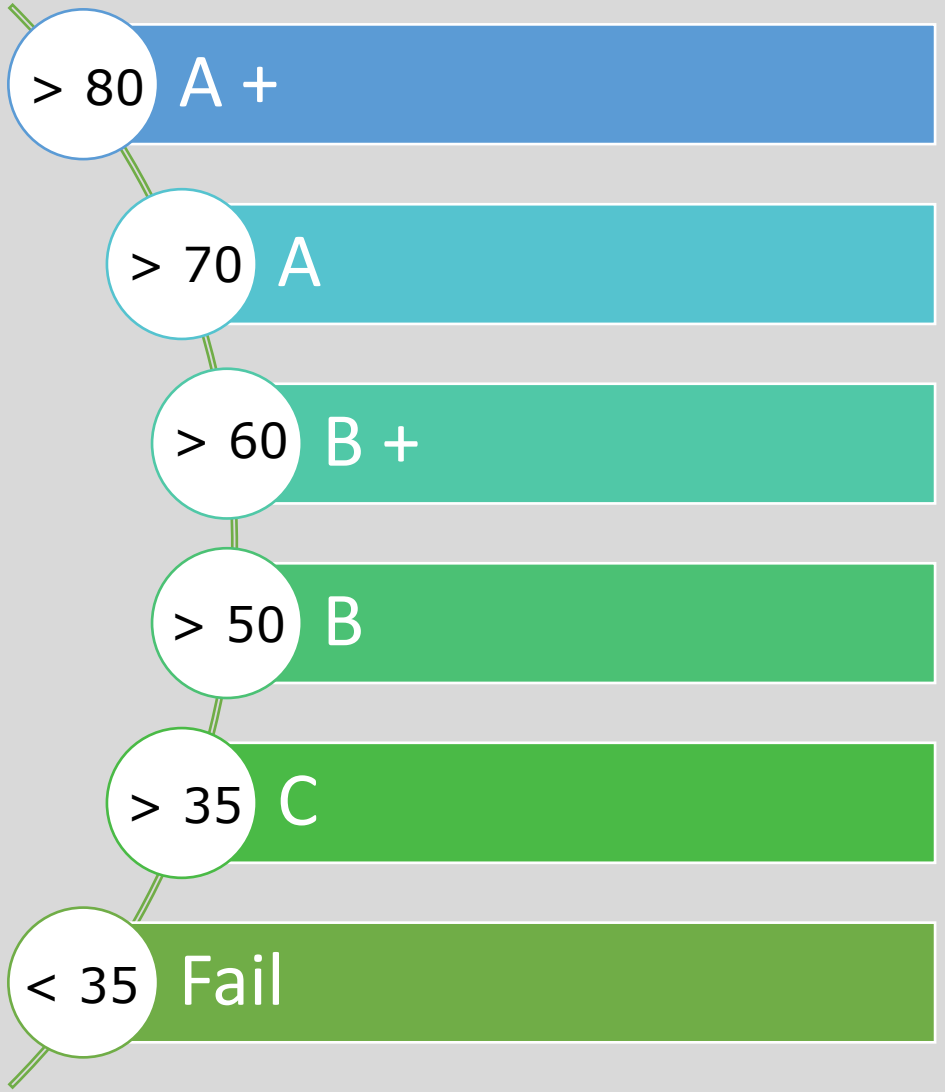
Watch Now

Conditional Statements



Conditional Statements

Student Marks



Conditional Statements

- if statement
- if-else statement
- if-else-if ladder
- Nested-if statement
- Switch statement

if statement

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

if – else statement

```
if(condition) {
        // statements..
} else {
        // statements..
}
```

if - else - if ladder

```
if(condition 1) {
        // statements..
} else if(condition 2) {
        // statements..
} else if(condition 3) {
        //statements..
} else {
        //statements..
}
```

Nested - if statement

```
if(condition 1) {
        if(condition 2) {
                //statements..
        }
}
```

switch statement

```
switch(condition) {
        case 1: statement 1;
                break;
        case 2: statement 2;
                break;
        ….
        case N: statement N;
                break;
        default: statements;
}
```

Code:

```
#include <stdio.h>

int main(){
    int marks;
    printf("Enter student marks ");
    scanf("%d", &marks);

    if(marks > 35){
        printf("Pass");
                    if(marks > 90) {
                                printf(" With Distinction");
                    }
    } else {
        printf("Fail");
    }
```

contd..

Code:

```
if(marks > 80){
        printf("Grade is A+");
    } else if(marks > 70){
        printf("Grade is A");
    } else if(marks > 60){
        printf("Grade is B+");
    } else if(marks > 50){
        printf("Grade is B");
    } else if(marks > 35) {
        printf("Grade is C");
    } else {
        printf("Fail");
    }
```

contd..

Code:

```
switch(marks/10){
        case 9: printf("Grade is A++");
                break;
        case 8: printf("Grade is A+");
                break;
        case 7: printf("Grade is A");
                break;
        case 6: printf("Grade is B+");
                break;
        case 5: printf("Grade is B");
                break;
        case 4: printf("Grade is C");
                break;
        default: printf("Fail");
    }


    return 0;
}
```

KALIYONA

Programming in C

Control Loops

Explained in Kannada

ಕನ್ನಡದಲ್ಲಿ

Watch Now

Control Loops

- Repeat same steps many number of times
- while loop
- do-while loop
- for loop

while

- Checks Condition first and then executes the Statements
- Used when number of iterations are not known

```
while(condition)
        // statements..
}
```

do-while

- Executes the statement first and

  then checks for the condition

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

for

- Most popular Loop Statement
- Used when number of iterations are known

```
for(initialisation; condition; increment/decrement) {
        statements…
}
```

Code:

```
#include <stdio.h>

int main()
{
    //Printing repeatedly
    printf("Welcome to Kaliyona\n");
    printf("Welcome to Kaliyona\n");
    printf("Welcome to Kaliyona\n");
    printf("Welcome to Kaliyona\n");
    printf("Welcome to Kaliyona\n");

    int i =1;
    //While Loop
    while(i<=5){
        printf("Welcome to Kaliyona : %d\n", i);
        i++;
    }
```

contd..

Code:

```
    //Do-While Loop
    do {
        printf("Welcome to Kaliyona : %d\n", i);
        i++;
    }while(i<=5);

    //For Loop
    //for(int i=0; i<5; i++){
    for(int i=5;i>0;i--){
        printf("Welcome to Kaliyona\n");
    }

        return 0;
}
```

KALIYONA

Programming in C

Arrays

Explained in Kannada

ಕನ್ನಡದಲ್ಲಿ

Watch Now

Arrays

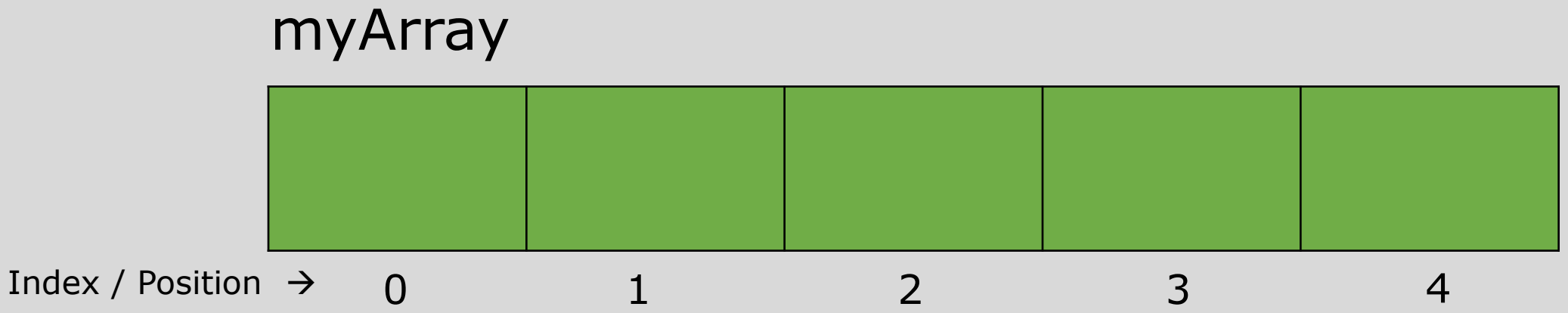
- Arrays are used to store multiple values in a single variable
- Syntax:

```
datatype arrayName[arraySize];
```

- Example

```
int myArray[5];
```

Arrays

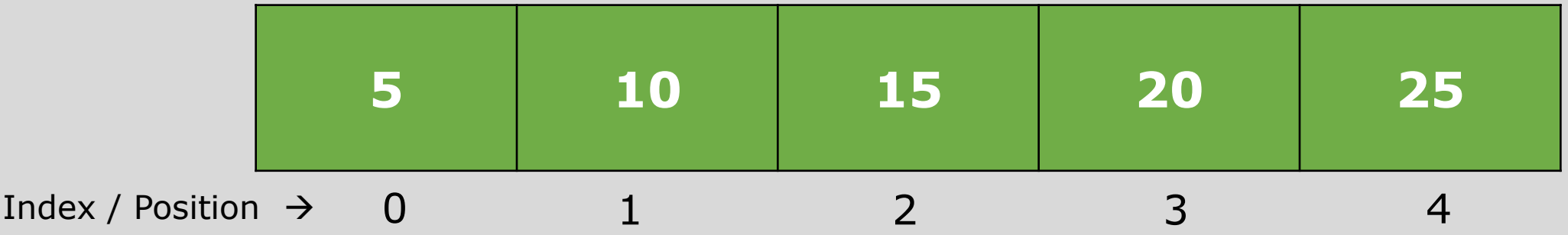


- Array Index always starts from 0

Accessing Array Elements

```
myArray[0] = 5;
myArray[1] = 10;
myArray[2] = 15;
myArray[3] = 20;
myArray[4] = 25;
```

myArray



Code:

```
#include <stdio.h>

int main()
{
    // int num1 = 5;
    // int num2 = 10;
    // int num3 = 15;
    // int num4 = 20;
    // int num5 = 25;

    //declaring arrays and accessing its elements
    int nums[5];
    nums[0] = 5; //index starts with zero
    nums[1] = 10;
    nums[2] = 15;
    nums[3] = 20;
    nums[4] = 25;

    //declaration with initialisation
    int nums[5] = {5, 10, 15, 20, 25};
```

contd..

Code:

```
//input and output with arrays
int nums[5], i;
printf("Enter first number : ");
scanf("%d", &nums[0]);
printf("\nEnter second number : ");
scanf("%d", &nums[1]);
printf("\nEnter third number : ");
scanf("%d", &nums[2]);
printf("\nEnter fourth number : ");
scanf("%d", &nums[3]);
printf("\nEnter fifth number : ");
scanf("%d", &nums[4]);

printf("\nArray elements are: \n");
printf("%d\n", nums[0]);
printf("%d\n", nums[1]);
printf("%d\n", nums[2]);
printf("%d\n", nums[3]);
printf("%d\n", nums[4]);
```

contd..

Code:

```
//using for loop
    printf("Enter 5 numbers: ");
    for(i=0; i<5; i++){
        scanf("%d", &nums[i]);
    }

    printf("Array Elements are : ");
    for(i=0;i<5;i++){
        printf("%d\t", nums[i]);
    }
    return 0;
}
```

KALIYONA

Programming in C

Strings

Explained in Kannada

ಕನ್ನಡದಲ್ಲಿ

Watch Now

Strings

- Strings are used to store series of characters
- C does not have a datatype called String
- Character array is used to define Strings in C
- '\0' defines null terminating character (end of the string)
- Format specifier for String is '%s'

```
char myStr[] = {'K', 'A', 'L', 'I', 'Y', 'O', 'N', 'A', '\0'};

char myStr[] = "KALIYONA";
```

String Functions

Function	Description
strlen	Get the length of the string
strcat	Concatenates two strings
strcpy	Copy one string value to another
strcmp	Compares two strings

Code:

```
#include <stdio.h>
#include <string.h>

int main()
{
    //char myStr[] = {'K', 'A', 'L', 'I', 'Y', 'O', 'N', 'A', '\0'};
    char myStr[] = "KALIYONA";

    //Printing Strings
    printf("%s", myStr);
```

contd..

Code:

```
//Using String in Loops
 int num = 0;

 for(int i=0; i<8; i++){
     if(myStr[i] == 'A'){
       num ++;
     }
     printf("%c\t", myStr[i]);
 }
 printf("\nCharacter A is occuring %d times in the given string", num);
```

contd..

Code:

```
//strlen
int length = strlen(myStr);
printf("length of the string is %d\n", length);
for(int i=0;i<length; i++){
    printf("%c", myStr[i]);
}

//strcpy
char str2[10];
strcpy(str2, myStr);
printf("\nCopied String value is %s", str2);
```

contd..

Code:

```
    //strcmp - returns 0 if string values matches else returns integer
    if(!strcmp(myStr, str2)){
        printf("\nStrings are equal\n");
    }

    //strcat
    char greeting[30] = "Welcome to ";
    strcat(greeting, myStr);
    printf("%s", greeting);

    return 0;
}
```

KALIYONA

Programming in C

Functions

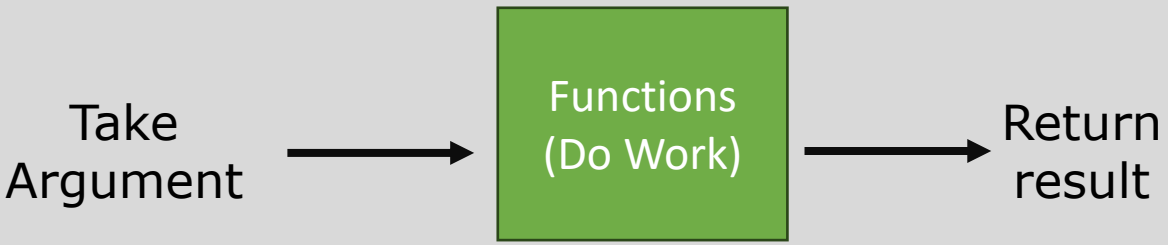
Explained in Kannada

ಕನ್ನಡದಲ್ಲಿ

Watch Now

Functions

- Block of code that performs a specific task
- Reusable code – define once, use many times
- 3 imp things to know
  - Function Declaration
  - Function Body
  - Calling a function



Function Declaration

Syntax:

```
returnType functionName(parameters);
```

Example:

```
void printMessage();
```

Function Body

Syntax:

```
returnType functionName(parameters) {

        //body of the function

}
```

Example:

```
void printMessage() {

        printf("Namaskara");

}
```

Calling / Invoking a function

Syntax:

```
functionName(parameter value);
```

Example:

```
printMessage();
```

Recursion

- Recursion is the technique of making a function call itself

Example:

```
int sum(int k) {
  if (k > 0) {
    return k + sum(k - 1);
  } else {
    return 0;
  }
}
```

Code:

```
#include <stdio.h>

//void return type does not return anything
void printMessage(char name[]);

//function with parameters and returning value
int add(int a, int b){
    return a+b;
}
```

contd..

Code:

```
//recursive function
/*
   sum(9);
   9 + sum(8)
   9 + 8 + sum(7)
   9 + 8 + 7 + sum(6)
   ....
*/
int sumOfseries(int n){
   if(n > 0){
      return n + sumOfseries(n-1);
   } else {
      return 0;
   }
}
```

contd..

Code:

```
int main()
{
    printMessage("Rahul");
    printf("\nSum of 5 and 4 is %d", add(5,4));
    printf("\nSum of 15 and 14 is %d", add(15,14));
    printf("\nSum of 30 and 40 is %d", add(30,40));
    printf("\n\n");
    printf("Sum of series until 9 is %d", sumOfseries(9));
    return 0;
}

void printMessage(char name[]){
    printf("Namaskara %s", name);
}
```

KALIYONA

Programming in C

Structures

Explained in Kannada

ಕನ್ನಡದಲ್ಲಿ

Watch Now

Structures

- User defined type
- Group several related variables into one place
- Also called as structs
- Each variable is known as a member of the structure
- ‘struct’ keyword is used to define Structures in C
- Dot(.) operator is used to access members of the structure

Structure Declaration

Syntax:

```
struct structureName {

        //members

};
```

Example:

```
struct student {

        int id;

        int age;

        char name[30];

};
```

Accessing Structure members

Example:

```
struct student s1;

s1.id = 101;

s1.age = 16;
```

Strings in Structures

- String variables in structures cannot be assigned value directly.
- Instead, use strcpy( ) function

Example:

```
strcpy(s1.name, “Rahul”);
```

Code:

```
#include <stdio.h>
#include <string.h>

struct student {
    int id;
    int age;
    char name[30];
};

int main()
{
    struct student s1;

    s1.id = 101;
    s1.age = 16;
    // s1.name = "Rahul"; -- results in error
    strcpy(s1.name, "Rahul");
```

contd..

Code:

```
//defining another structure
   struct student s2;
   s2.id = 102;
   s2.age = 17;
   strcpy(s2.name, "Ankita");

   printf("Id, name and Age of student 1 is %d, %s and %d \n", s1.id, s1.name, s1.age);
   printf("Id, name and Age of student 2 is %d, %s and %d", s2.id, s2.name, s2.age);

   return 0;
}
```

KALIYONA

Programming in C

Pointers

Explained in Kannada

ಕನ್ನಡದಲ್ಲಿ

Watch Now

Pointers

Pointer is a variable which stores the memory address of another variable as its value

- '*' operator is used to create pointers
- '&' is called the reference operator

Example:

```
int age = 18;

Int *ptr = &age;
```

Pointers

Example:

```
int age = 18;

Int *ptr = &age;
```

Pointers & Functions

- Call by Value
  - Call a function by passing parameters value
- Call by Reference
  - Call a function by passing address of the variables

Code:

```
#include <stdio.h>

//call by value
void swap(int x, int y){
    int t= x;
    x=y;
    y=t;
    printf("Swapped integers are %d and %d\n", x,y);
}

//call by reference
void swapByReference(int *x, int *y){
    int t= *x;
    *x = *y;
    *y = t;
    printf("Swapped integers are %d and %d\n",
*x,*y);
}
```

contd..

Code:

```
int main() {

    int age = 20;
    int* ptr = &age; //stores address of age
    int **pptr = &ptr; //stores address of ptr

    printf("Value of age is %d", age);
    printf("\nAddress of age is %p", ptr);

    printf("\nValue of age using pointer is %d", *ptr);
    printf("\nAddress of pointer variable is %p", &ptr);

    printf("\nValue of age using double pointers is %d", **pptr);

                                                        contd..
```

Code:

```
    int a = 5, b=10;
    swap(a,b);
    printf("Values are now %d and %d\n", a,b); //numbers are not swapped

    swapByReference(&a,&b);
    printf("Values are now %d and %d\n", a,b); //numbers are swapped

    return 0;
}
```

KALIYONA

Programming in C

File Operations

Explained in Kannada

ಕನ್ನಡದಲ್ಲಿ

Watch Now

Creating a File

- File Operations can be performed by declaring a pointer of type FILE
- fopen( ) function is used to create or open a file
- File modes
    - w → write mode
    - a → append mode
    - r → read mode

Writing a File

- fprintf( ) function is used to write contents to a file

```
fprintf(fptr, "Some text");
```

Reading a File

- fgets( ) function is used to read contents of a file

```
fgets(myString, 100, fptr);
```

Code:

```
#include <stdio.h>

int main() {

  FILE *fptr;
  char myString[100];

  // Create a file on your computer (kaliyona.txt)
  fptr = fopen("kaliyona.txt", "w"); //write mode
  fptr = fopen("kaliyona.txt", "a"); //append mode
  fptr = fopen("kaliyona.txt", "r"); //read mode
```

Contd..

Code:

```
  fprintf(fptr, "Namaskara"); //Writes contents to the file
  fprintf(fptr, " Kaliyona Family");

  //fgets(myString, 100, fptr); //gets only first line
  //printf("%s", myString);

  //use while loop to print all lines
  while(fgets(myString, 100, fptr)){
      printf("%s", myString);
  }

  // Close the file
  fclose(fptr);

  return 0;
}
```