# C Language Sheet

Loading...

## Basics

Basic syntax and functions from the C programming language.

### Boilerplate Code

#include<stdio.h>int main(){return(0);}

Copy

### printf function

It is used to show output on the screen

printf("Hello World!")

Copy

### scanf function

It is used to take input from the user

scanf("placeholder", variables)

Copy

## Comments

A comment is the code that is not executed by the compiler, and the programmer uses it to keep track of the code.

### Single line comment

// It's a single line comment

Copy

### Multi-line comment

/\* It's a

multi-line

comment

\*/

Copy

## Data types

The data type is the type of data

### Character type

Typically a single octet(one byte). It is an integer type

char variable\_name;

Copy

### Integer type

The most natural size of integer for the machine

int variable\_name;

Copy

### Float type

A single-precision floating-point value

float variable\_name;

Copy

### Double type

A double-precision floating-point value

double variable\_name;

Copy

### Void type

Represents the absence of the type

void

Copy

## Escape Sequences

It is a sequence of characters starting with a backslash, and it doesn't represent itself when used inside string literal.

### Alarm or Beep

It produces a beep sound

\a

Copy

### Backspace

It adds a backspace

\b

Copy

### Form feed

\f

Copy

### Newline

Newline Character

\n

Copy

### Carriage return

\r

Copy

### Tab

It gives a tab space

\t

Copy

### Backslash

It adds a backslash

\\

Copy

### Single quote

It adds a single quotation mark

\'

Copy

### Question mark

It adds a question mark

\?

Copy

### Octal No.

It represents the value of an octal number

\nnn

Copy

### Hexadecimal No.

It represents the value of a hexadecimal number

\xhh

Copy

### Null

The null character is usually used to terminate a string

\0

Copy

## Conditional Instructions

Conditional statements are used to perform operations based on some condition.

### If Statement

if (/\* condition \*/){/\* code \*/}

Copy

### If-else Statement

if (/\* condition \*/){/\* code \*/}else{/\* Code \*/}

Copy

### if else-if Statement

if (condition) {// Statements;}else if (condition){// Statements;}else{// Statements}

Copy

### Switch Case Statement

It allows a variable to be tested for equality against a list of values (cases).

switch (expression) {case constant-expression:

statement1;

statement2;break;case constant-expression:

statement;break;...default:

statement;}

Copy

## Iterative Statements

Iterative statements facilitate programmers to execute any block of code lines repeatedly and can be controlled as per conditions added by the programmer.

### while Loop

It allows execution of statement inside the block of the loop until the condition of loop succeeds.

while (/\* condition \*/){/\* code \*/}

Copy

### do-while loop

It is an exit controlled loop. It is very similar to the while loop with one difference, i.e., the body of the do-while loop is executed at least once even if the expression is false

do{/\* code \*/} while (/\* condition \*/);

Copy

### for loop

It is used to iterate the statements or a part of the program several times. It is frequently used to traverse the data structures like the array and linked list.

for (int i = 0; i < count; i++){/\* code \*/}

Copy

### Break Statement

break keyword inside the loop is used to terminate the loop

break;

Copy

### Continue Statement

continue keyword skips the rest of the current iteration of the loop and returns to the starting point of the loop

continue;

Copy

## Functions & Recursion

Functions are used to divide an extensive program into smaller pieces. It can be called multiple times to provide reusability and modularity to the C program.

### Function Definition

return\_type function\_name(data\_type parameter...){ //code to be executed }

Copy

### Recursion

Recursion is when a function calls a copy of itself to work on a minor problem. And the function that calls itself is known as the Recursive function.

void recurse(){... .. ...recurse();... .. ...}

Copy

## Pointers

Pointer is a variable that contains the address of another variable,

### Declaration

datatype \*var\_name;

Copy

## Arrays

An array is a collection of data items of the same type.

### Declaration

data\_type array\_name[array\_size];

Copy

### Accessing element

int variable\_name = array[index];

Copy

## Strings

A string is a 1-D character array terminated by a null character ('\0')

### Declaration

char str\_name[size];

Copy

### gets() function

It allows you to enter multi-word string

gets("string");

Copy

### puts() function

It is used to show string output

puts("string");

Copy

### String Functions strlen()

It is used to calculate the length of the string

strlen(string\_name);

Copy

### strcpy() function

It is used to copy the content of second-string into the first string passed to it

strcpy(destination, source);

Copy

### strcat() function

It is used to concatenate two strings

strcat(first\_string, second\_string);

Copy

### strcmp() function

It is used to compare two strings

strcmp(first\_string, second\_string);

Copy

## Structures

The structure is a collection of variables of different types under a single name. Defining structure means creating a new data type.

### Structure syntax

struct structureName {

dataType member1;

dataType member2;...};

Copy

### typedef keyword

typedef function allows users to provide alternative names for the primitive and user-defined data types.

typedef struct structureName {

dataType member1;

dataType member2;...}new\_name;

Copy

## File Handling

A set of methods for handling File IO (read/write/append) in C language

### FILE pointer

FILE \*filePointer;

Copy

## 

### Opening a file

It is used to open file in C.

filePointer = fopen(fileName.txt, w)

Copy

### fscanf() function

It is used to read the content of file.

fscanf(FILE \*stream, const char \*format, ...)

Copy

### fprintf() function

It is used to write content into the file.

fprintf(FILE \*fptr, const char \*str, ...);

Copy

### fgetc() function

It reads a character from a file opened in read mode. It returns EOF on reaching the end of file.

fgetc(FILE \*pointer);

Copy

### fputc() function

It writes a character to a file opened in write mode

fputc(char, FILE \*pointer);

Copy

### Closing a file

It closes the file.

fclose(filePointer);

Copy

## Dynamic Memory Allocation

A set of functions for dynamic memory allocation from the heap. These methods are used to use the dynamic memory which makes our C programs more efficient

### malloc() function

Stands for 'Memory allocation' and reserves a block of memory with the given amount of bytes.

ptr = (castType\*) malloc(size);

Copy

### calloc() function

Stands for 'Contiguous allocation' and reserves n blocks of memory with the given amount of bytes.

ptr = (castType\*)calloc(n, size);

Copy

### free function

It is used to free the allocated memory.

free(ptr);

Copy

### realloc() function

If the allocated memory is insufficient, then we can change the size of previously allocated memory using this function for efficiency purposes

ptr = realloc(ptr, x);

Copy