C PROGRAMMING CHEATSHEET

C Program basic structure

```c
#include <stdio.h>

int main ()
{
  // Your code here
  return 0;
}
```

printf function

It is used to show output on the screen

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

scanf function

It is used to take input from the user

```c
scanf("format_specifier", &variables)
```

Comments

Comments are ignored by the compiler and do not affect the execution of the program.

Single Line Comments

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

Multi-line comments

```c
/* This is a
multi line
comment */
```

Variables

Variables are containers for storing data values.

- **In C, variables are classified into different types:**
- **int:** an integer variable can store only an integer.
- **float:** a floating point variable can store only a floating point numbers.
- **char:** a character variable can store only a character.

Variable Declaration

Variable declaration is the process of specifying the name and type of a variable before using it in the program.

```
// Declare a variable
int myNum;

// Assign a value to the variable
myNum = 15;
```

Data Types

A data type in C specifies the type of data that a variable can store.

C provides several built-in data types, including:

Data Type	Size	Description
int	2 or 4 bytes	Stores whole numbers, without decimals
float	4 bytes	Stores fractional numbers, containing one or more decimals.
char	1 byte	Stores a single character/letter/numbers
double	8 bytes	Stores fractional numbers, containing one or more decimals.

```c
#include <stdio.h>
int main()
{
 int integer = 20;
    float floating = 10.32;
    char character = 'B';

    printf("%d\n", integer);
    printf("%f\n", floating);
    printf("%c\n", character);
    return 0;
}
```

Format Specifiers

Format Specifier	Type
%c	Character
%d	Signed integer
%f	Float values
%i	Unsigned integer
%l or %ld or %li	Long
%lf	Double
%o	Octal representation
%p	Pointer
%s	String
%u	Unsigned int

Escape Sequences

Escape sequence	Meaning
\\	\ character
\'	' character
\b	Backspace
\f	Form feed
\n	Newline
\t	Horizontal tab
\v	Vertical tab

if...else Statements

```
if (condition) {
  //code to be executed if the condition is true
} else {
  //code to be executed if the condition is false
}
```

if-else Ladder

```
if (condition) {
  //code
}
else if (condition) {
  //code
}
else if (condition) {
  //code
}
else {
  //code
}
```

Switch Statement

```
switch (integer expression)
{
case 1:
do this;

case 2:
do this;

case 3:
do this;

default:
do this;
}
```

While Loop

The while loop is used when you want to execute a block of code as long as a condition is true.

```
while (condition)
{
    // block of code
}
```

for Loop

A for loop in C programming is a repetition control structure that allows programmers to write a loop that will be executed a specific number of times.

```
for (initialization; testExpression; increment/decrement)
{
    // block of code
}
```

do...while Loop

The do-while loop is similar to the while loop. This loop would execute its statements at least once, even if the condition fails for the first time.

```
do {
  // block of code
} while (condition);
```

Break Statement

The break statement is used to break out of the loop in which it is encountered. The break statement is used inside loops or switch statements in C programming.

```
#include <stdio.h>

int main () {
  int i;
  for (i = 0; i < 10; i++) {
    if (i == 6) {
      break;
    }
    printf ("%d", i);
  }
  return 0;
}
```

Continue Statement

The continue statement skips the loop's current iteration and proceeds to the next one.

```c
#include <stdio.h>

int main () {
  int i,j;
  for (i = 1; i <= 2; i++)
  {
    for (j = 1; j <= 2; j++) {
      if (i == j)
       continue;
      printf ("%d%d", i,j);
    }
  }
  return 0;
}
```

Arrays

An array is a collection of similar data items. It is stored at contiguous memory locations in arrays.

Array Declaration

```c
data_type array_name [size];
```

Accessing Elements of an Array

```c
#include <stdio.h>

int main()
{
    int myArr[] = {2, 6, 8, 10};

    printf("%d", myArr[2]);
    return 0;
}
```

Strings

Strings are used for storing text/characters. For example, "Hello World" is a string of characters.

```
char name[] = "John";
```

String Declaration

```
char string_name[string_size];
```

strlen() function

This function counts the number of characters present in a string.

Example:

```
#include <stdio.h>

int main () {
  char str1[] = "Tutorials4Coding";
  printf("%d", strlen(str1));

  return 0;
}
```

strcpy():

This function copies the contents of one string to another.

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

int main () {
  char str1[20] = "Tutorials4Coding";
  char str2[20];

  strcpy(str2, str1);

  printf("%s", str2);

  return 0;
}
```

strcat():

This function concatenates the source string at the end of the target string.

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

int main () {
  char str1[20] = "Hello ";
  char str2[] = "World";

  strcat(str1, str2);

  printf("%s", str1);

  return 0;
}
```

Functions

Functions in C are blocks of code that perform a specific task. They are intended to improve code readability, modularity, and reusability by dividing a program into parts.

Function Declarations

```c
return_type function_name (para_1, para_2){

  // block of the function

}
```

Function Call

A function is called by its name, followed by parenthesis.

```c
#include <stdio.h>

// Function declaration
void func();

int main() {
  func(); // calling the function
  return 0;
}

// Function definition
void func() {
    printf("Execution Succesfull.");
}
```

Recursion

Recursion is a programming method in C that involves calling a function on itself to solve a problem.

```c
#include <stdio.h>
int fibo(int);
int main()
{
    int terms = 12, i, n = 0;
    for (i = 1; i <= terms; i++)
    {
        printf ("%d\t", fibo (n));
        n++;
    }
    return 0;
}

int fibo(int n)
{
    if (n == 0 || n == 1)
        return n;
    else
        return (fibo(n - 1) + fibo (n - 2));
}
```

Structures

Structures are used to group variables of various data types under a single name. For Example, a 'book' is a collection of items like title, author, publisher, number of pages, date of publication etc. for dealing with such data C provides a data type called structure.

```
struct MyStructure {
  dataType member1;
  dataType member2;
};
```

Files

File handling in C is a fundamental part of programming that lets you read and write files on the system. It consists of opening files, reading data from them, writing data to them, and then closing them once completed.

Creating a File

To create a file in C, use the `fopen()` function with the proper mode.

```
FILE *fptr;
fptr = fopen("filename.txt", "w");
```

Open a File

To open a file, use the `fopen()` function.

```
FILE* fopen("fileopen","mode");
```

Reading From a File

Use the functions like `fscanf(),fgets()` to read data from the file.

```
FILE *fptr;
fptr = fopen("filename.txt", "r");
```

Closing a File

When we have finished reading from the file, we need to close it. This is done using the fclose() function.

```
fclose(fp);
```