C Cheatsheet

Download PDF

Download Link : https://github.com/rohan-kiratsata/coding-cheatsheets/tree/main/PDFs

Table of content

Boilerplate Code

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

printf() function

It is used to print anything/show output on console

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

scanf() function

It is used to take input from user

```
scanf("format specifier",variable_name);
```

Variables

It is data name that is used to store the data value in the memory.

Rules to declare variables:

Variable Name	Valid?	Why?

Variable Name	Valid?	Why?
int	Not Valid	keywords are not allowed
amount$	Not Valid	Dollar Sign is not allowed
your name	Not Valid	Space between variable name is not allowed
average_score	Valid	underscore can be used as space
First_name	Valid	-
int_type	Valid	Keywords can be used as combination

Basic Data Types

char - It stores single character. Size - 1 byte

```
char var_name = 'a';
// Another example
char var_name2 = 'x';
```

int - It stores an integer value. Size - 4 bytes

```
int age = 18;
//Another example
int amount = 10000;
```

float - It stores an floating point value with 6 digit precision. Size - 4 bytes

```
float radius = 5.8;
//Another example
float area = 15.2;
```

double - It stores an float value with 14 digit precision. Size - 8 bytes.

```
double var = 12.1531452;
//Another Example
double more_var = 542.12452;
```

void - Represents the absence of type.

```
void main(){
    //does not return anything.
}
```

Format Specifiers

Format Specifier	Type
%c	Character
%d	Integer
%f	float
%lf	double

Format Specifier	Type
%l	long
%Lf	long double
%lld	long long
%o	octal representation
%p	pointer
%s	string
%%	prints % symbol

Escape Sequences

Escape Sequence	Type
\a	Produces Alarm/Beep Sound
\b	Backspace
\f	Form Feed
\n	New Line
\t	Tab Space
\v	Tab Space - Vertically
\\	Backslash
\"	Double Quote
\'	Single Quote
\?	Question Mark

Operators

Arithmetic Operators

Operators	Example	Meaning
+	a+b	Addition or unary plus
-	a-b	Subtraction or unary minus
*	a*b	Multiplication
/	a/b	Division
%	a%b	Modulo Division-Gives remainder

Relational Operators

Operator	Example	Meaning
<	a < b	is less than

Operator	Example	Meaning
>	a > b	is greater than
<=	a<=b	is less than or equal to
>=	a>=b	is greater than or equal to

Logical Operators

Operator	Example	Meaning
&&	a && b	logical AND
||	a || b	logical OR
!	a ! b	logical NOT

Learn more about logic gates here

Increment and Decrement Operators

`variable_name++` Here `++` is a increment operator, it increments the value of variable by 1. `variable_name--` Here `--` is a decrement operator and it decrements the value of variable by 1.

More examples:

```
int a = 1; // a value is 1
a++;       // Now Value becomes 2

int b = 5; // b value is 5
b--;       // Now, b value is 4
```

Conditional Statements

If Statement

```
if(codition)
{
    //statements or code
}
//example
int a = 1, b = 5;
if(a < b){
    printf("A is smaller");
}
```

If-Else Statements

```
if(condition){
    //statements
}
else{
    //statements
}
//example
int a=1; b=5;
if(a < b){
    printf("A is smaller");
}
else{
    printf("B is smaller");
}
```

If Else-If Statements

```
if(condtion){
    //code
}
else if(another_condtion){
    //code
}
else{
    //code
}
```

Switch Case Statements

```
switch(expression)
{
    case constant-expression:
        statement1;
        statements2;
        break;
    case constant-expression:
        another_statements;
        break;

    .....(n number of cases)
    default;
        statements;
}
```

Iterative Statements

It executes the statements inside a block of loop until condition is false

While loop

```
while(condition){

    //code
}
//example

int a = 10;
while(a <=10){
    printf("%d",a);
}//will print 1 to 10
```

Do-While Loop

```
do{
    // code
}
while(codition);
```

For Loop

```
for(int i=0; i < counter; i++){
    // code
}
```

Break Statement

Break keyword is used to terminate the loop.

```
break;
```

Continue Statement

Continue keyword skips the rest of iterative code of loop and return to starting point of loop.

```
continue;
```

Functions

Functions are used to divide the code and to avoid repetitive task. It provides reusability and readability to code. **Function Declaration**

```
return_type function_name(data_type-parameters){
    //code
}
//  Example of function to add two numbers

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

Recursion

Recursion is the process of repeating items in a self-similar way.If a program allows you to call a function inside the same function, then it is called a recursive call of the function.

```
void myFunction(){
    myFunction();   //Function calling itself
}
```

```
//Factorial Using Recursion
long factorial(long n){
    if(n==0){
        return 1;
    }
    return n * factorial(n -1);
}

int main(){
    int n = 5;
    printf("Factorial of %d is %l.",n,factorial(n));
    return 0;
}
//OUTPUT : Factorial of 5 is 120.
```

Pointers

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

```
datatype *var_name;
```

Arrays

Array is an collection of data of same data-type.

Declaration

```
data_type array_name[array_size];
```

Fetching Array Element Array index starts from 0.

```
data_type variable_name = arr[index]
```

```
//Example

int arr = {1,2,3,4,5};
printf("%d",arr[0])
//OUTPUT : 1
```

Strings

It is basically 1D character array. It character of string is null character (\0) **Declaration**

```
char string_name[size];
```

Important String Functions

gets() function It is used to take input of multi-character string

```
gets("string");
```

puts() function It is used to show string output

```
puts("string");
```

strlen() function It prints the length of string.

```
strlen(string_name);
```

strcat() function It is used to concatenate two strings.

```
strcat(string1, string2);
```

strcmp() function It is used to compare two strings. Gives output in 0/1.

```
strcmp(string1, string2);
```

strcpy() function It is used to copy the content of 1st string to another string.

```
strcpy(string1, string2);
```

Structures

A structure creates a data type that can be used to group items of possibly different types into a single type. **Declaration**

```
struct student
{
    char name[50];
    int class;
    float percentage;
    char college[50];
};      //Notice the semicolon
```

Dynamic Memory Allocation

If you are aware of the size of an array, then it is easy and you can define it as an array. For example, to store a name of any person, it can go up to a maximum of 100 characters. But now let us consider a situation where you have no idea about the length of the text you need to store, for example, you want to store a detailed description about a topic. Here we need to define a pointer to character without defining how much memory is required and later. So we use Dynamic Memory Allocation.

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

```
var = (casting_type*)malloc(size);
//Example
var = (int*)malloc(n * sizeof(int))
```

calloc() function Stands for “contiguous allocation” method in C is used to dynamically allocate the specified number of blocks of memory of the specified type.

```
var = (cast_type*)calloc(n, size);
```

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

```
var = realloc(var2,n);
```

File Handling

Creating File Pointer

```
FILE *file
```

Opening a File

```
file = fopen(file_name.txt,w)
```

fscanf() function Used to read file content

```
fscanf(FILE *stream, const char *format, ..);
```

fprintf() function Used to write the file content

```
fprintf(FILE *var, const char *str,..);
```

Closing a File

```
fclose(file);
```

This Cheatsheet is contributed by Rohan Kiratsata

Personal Site Link : https://rohan-kiratsata.github.io

Something is missing? Contribute to this repo and improve.

Contribution Repo: https://github.com/rohan-k14/coding-cheatsheets

---