C Cheat Sheet

Matrix Codez

Comments

Single line comment

```
// this is a single-line comment
```

Multi-line comment

```
/* this is a multi-line comment

we can write multiple lines at once */
```

printf() and scanf()

```
printf("Hello welcome to Matrix Codez"); // it'll display output on the console

scanf("format specifier", variables); // to read input from user

int num;

scanf("%d", &num);
```

Escape Sequences

/b	backspace
/f	form feed
/n	creates a new line
/r	carriage return
/t	horizontal tab
/"	double quote
/'	single quote
/v	vertical tab
/a	to alert
/0	to terminate a string

Data types

Integer

int variable_name;

Float

float variable_name;

Char

char variable_name;

Double

double variable_name;

Void

void

Variables

Valid Variables	Invalid Variables
result	2number
sum	int, float //we can't use keywords as variables
_average	num+ //special character
num1	

Operators

Arithmetic Operators

+

// to add

For example int sum = x + y;

-

// to subtract

For example int subtract = x - y;

*

// to multiply

For example int product = x * y;

/

// Division

For example int division = x / y;

%

// Modulus this operator returns the remainder value

For example int modulus = x % y;

Relational Operators

// these operators the relation between two variables

// used to compare the values of two variables

Operator	Example	Meaning
>	a > b	a is greater than b
<	a < b	a is less than b
>=	a >= b	a is greater than or equal to b
<=	a <=b	a is less than or equal to b
==	a == b	a is equal to b
!=	a !=b	a is not equal to b

For logical operators and bitwise operators look at our c tutorial

Precedence and Associativity rules

Operator	Priority	Associativity
{}, (), []	first	left to right
++,--,!	second	right to left
*,/,%	third	left to right
+,-	fourth	left to right
<,<=,>, >=,==,!=	fifth	left to right
&&	sixth	left to right
||	seventh	left to right
?	eighth	right to left
=,+=,-=, *=,/=,%=	ninth	right to left

Conditional Statements

If Statement

```
if ( condition) {
code  block
}
```

If-else Statement

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

if else-if

```
if (condition) {

// Statements;

} else if (condition){

// Statements;

}

else

{

// Statements

 }
```

Switch Case Statement

```
switch(expression)

{

case value1:

   block1:

   break;

case value2:

    block2;

    break;

… … … … … …

default:

     default block;

     break;

}

statement;
```

Iterative statements or Loops

while loop

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

do - while loop

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

for loop

```
 for (int i = 0; i < count; i++)
 {
code  block
}
```

Functions

Function Definition

```
 return_type function_name(parameter list)
{
The code block for the execution of our task
 }
```

Recursion

```
void recurse()

{

 ... .. ...

recurse();

 ... .. ... }
```

Arrays

Declaration

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

To Access an Element from an array

```
int variable_name = array[index];  // index is start with 0
```

Strings

Declaration

```
// in c we don’t have “strings” we use an array of characters as a string

char str_name[size];
```

gets() function

```
gets("Hello");
```

puts() function

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

strlen()

```
strlen(string_name);
```

Structures

Syntax

```
 struct structureName
 {
 dataType member1;
 dataType member2;
 ...
dataType member n;
};
```

Union

Syntax

```
union unionName
 {
 dataType member1;
 dataType member2;
 ...
dataType member n;
};
```

Dynamic Memory Allocation

malloc()

ptr = (castType*) malloc(size);

calloc()

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

realloc()

ptr = realloc(ptr, x);

File Handling

Opening a file

filePointer = fopen(fileName.txt, w);

fscanf()

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

fprintf()

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