Cheatography

C info and some basic characteristics Cheat Sheet

by EstudianteUpy (Estudianteupy) via cheatography.com/124029/cs/23631/

C Cheat sheet

Basic Types	Arithmetic types and are further classified into: (a) integer types and (b) floating-point types.
Enumerated types	Arithmetic types and they are used to define variables that can only assign certain discrete integer values throughout the program.
The type void	Type specifier void indicates that no value is available.
Derived types	Include (a) Pointer types, (b) Array types, (c) Structure types, (d) Union types and (e) Function types.

Data types

Integer Types	Storage size	Value range	
char	1 byte	-128 to 127 or 0 to 255	%c
unsigned char	1 byte	0 to 255	
signed char	1 byte	-128 to 127	

Data types (cont)

int	2 or 4 bytes	-32,768 to 32,767 or -2,147,483,648 to 2,147,-483,647	%d or %i
unsigned int	2 or 4 bytes	0 to 65,535 or 0 to 4,294,-967,295	%u
short	2 bytes	-32,768 to 32,767	%hi
unsigned short	2 bytes	0 to 65,535	
long	8 bytes	-92233720368-54775808 to 922337203685-4775807	%li
unsigned long	8 bytes	0 to 184467440737095516 15	

Floating-Point Types	Storage size	Value range	
float	4 byte	1.2E-38 to 3.4E+38 (6DP)	%f
double	8 byte	2.3E-308 to 1.7E+308 (15DP)	%lf
long double	10 byte	3.4E-4932 to 1.1E+4932 (19DP)	%Lf
string	x50 char		%s

* DP = Decimal precision.

Operators

Operators	
||/|	Or
&&/&	And
==	Equal to
!	Not
!=	Non equal to

Arithmetic Operators	
+	plus
-	rest
/	divide
*	product
%	reminder
++/ --	Increasing/decreasing

Comparison	
<	lower than
<=	lower or equal than
>=	greater or equal than
>	greater than

Main Libraries and Functions

`<assert.h>`	Program assertion functions
`<ctype.h>`	Character type functions
`<locale.h>`	Localization functions
`<math.h>`	Mathematics functions
`<setjmp.h>`	Jump functions
`<signal.h>`	Signal handling functions

By **EstudianteUpy** (Estudianteupy)

cheatography.com/estudianteupy/

Published 3rd July, 2020.
Last updated 3rd July, 2020.

Main Libraries and Functions (cont)

<stdarg.h>	Variable arguments handling functions
<stdio.h>	Standard Input/Output functions
<stdlib.h>	Standard Utility functions
<string.h>	String handling functions
<time.h>	Date time functions

std functions:

rand()

Returns a (predictable) random integer between 0 and RAND_MAX based on the randomizer seed.

RAND_MAX

The maximum value rand() can generate.

srand(unsigned integer);

Seeds the randomizer with a positive integer.

(unsigned) time(NULL)

Returns the computer's tick-tock value. Updates every second.

i/o functions

scanf() and printf() functions

printf( )	returns the number of characters printed by it.
scanf()	returns the number of characters read by it.

getchar() & putchar() functions

getchar()	reads a character from the terminal and returns it as an integer.
putchar()	displays the character passed to it on the screen and returns the same character.

gets() & puts() functions

i/o functions (cont)

gets()	reads a line from `stdin`(standard input) into the buffer pointed to by str pointer, until either a terminating newline or EOF (end of file) occurs.
puts()	writes the `string` str and a trailing newline to *stdout*.

The standard input-output header file, named stdio.h contains the definition of the functions printf() and scanf(), which are used to display output on screen and to take input from user respectively.

ontrol structures and statements

Loop structure

A loop structure is used to execute a certain set of actions for a predefined number of times or until a particular condition is satisfied. There are 3 control statements available in C to implement loop structures. *While, Do while and For statements.*

The while statement

Syntax for while loop is shown below:

```
while(condition) // This condition is tested for TRUE or FALSE. Statements inside curly braces are executed as long as condition is TRUE
{
statement 1;
statement 2;
statement 3;
}
```

ontrol structures and statements (cont)

The condition is checked for TRUE first. If it is TRUE then all statements inside curly braces are executed. Then program control comes back to check the condition has changed or to check if it is still TRUE. The statements inside braces are executed repeatedly, as long as the condition is TRUE. When the condition turns FALSE, program control exits from while loop.

The do while statement

Syntax for do while loop is shown below:

```
do
{
statement 1;
statement 2;
statement 3;
}
`while(condition);
```

The for statement

Syntax of for statement is shown below:

```
for(initialization statements;test condition;iteration statements)
{
statement 1;
statement 2;
statement 3;
}
```

Control structures

Selective control structure

Selection structures are used to perform 'decision making' and then branch the program flow based on the outcome of decision making. Selection structures are implemented in C with If, If Else and Switch statements.

The syntax format of a simple if statement is as shown below.

```
if (expression) // This expression is evaluated. If expression is TRUE statements inside the braces will be executed
{
statement 1;
statement 2;
}
statement 1; // Program control is transfered directly to this line, if the expression is FALSE
statement 2;
```

Syntax format for If Else statement is shown below.

```
if(expression 1) // Expression 1 is evaluated. If TRUE, statements inside the curly braces are executed.
{ // If FALSE program control is transferred to immediate else if statement.
statement 1;
statement 2;
}
else if(expression 2)// If expression 1 is FALSE, expression 2 is evaluated.
```

Control structures (cont)

```
{
statement 1;
statement 2;
}
else if(expression 3) // If expression 2 is FALSE, expression 3 is evaluated
{
statement 1;
statement 2;
}
else // If all expressions (1, 2 and 3) are FALSE, the statements that follow this else (inside curly braces) is executed.
{
statement 1;
statement 2;
}
other statements;
```

Switch statement

Switch is a multi branching control statement. Syntax for switch statement is shown below.

```
switch(expression)// Expression is evaluated. The outcome of the expression should be an integer or a character constant
{
case value1: // case is the keyword used to match the integer/character constant from expression.
//value1, value2 ... are different possible values that can come in expression
statement 1;
statement 2;
break; // break is a keyword used to break the program control from switch block.
case value2:
statement 1;
statement 2;
break;
default: // default is a keyword used to execute a set of statements inside switch, if no case values match the expression value.
statement 1;
statement 2;
break;
}
```