C cheat sheet

Here is a cheat sheet for beginners in C programming that covers basic syntax, data types, operators, control structures, functions, pointers, and recursion:

Basic Syntax

C programs are made up of functions that contain statements. Each statement ends with a semicolon (;). The main function is the entry point of the program.

```
#include <stdio.h>int main() {
    printf("Hello, world!");
    return 0;
}
```

Data Types

C has several built-in data types, including:

- int (integer)
- float (floating-point number)
- double (double-precision floating-point number)
- char (character)
- bool (boolean)

```
int x = 10;
float y = 3.14;
double z = 2.71828;
char c = 'A';
bool b = true;
```

Operators

C has various operators for performing arithmetic, comparison, logical, and bitwise operations.

- Arithmetic operators: +, -, *, /, %
- Comparison operators: ==, !=, <, >, <=, >=
- Logical operators: && (and), || (or), ! (not)
- Bitwise operators: &, |, ^, ~, <<, >>

```
int x = 10, y = 20;
int z = x + y;          // 30
bool b = x < y && x > 0; // true
int a = x << 2;         // 40
```

Control Structures

C provides control structures for making decisions and repeating statements.

- If statement: executes a statement if a condition is true.

```
if (x > 0) {
    printf("x is positive");
}
```

- If-else statement: executes one statement if a condition is true and another statement if it is false.

```
if (x > 0) {
    printf("x is positive");
} else {
    printf("x is non-positive");
}
```

- For loop: repeats a statement a fixed number of times.

```
for (int i = 0; i < 10; i++) {
    printf("%d ", i);
}
```

- While loop: repeats a statement while a condition is true.

```
cCopy code
while (x > 0) {
    x--;
    printf("%d ", x);
}
```

- Do-while loop: repeats a statement at least once and then while a condition is true.

```
cCopy code
do {
    x--;
    printf("%d ", x);
} while (x > 0);
```

Functions

Functions are reusable blocks of code that perform a specific task.

```
int add(int x, int y) {
    return x + y;
}

int result = add(10, 20); // 30
```

Pointers

Pointers are variables that store memory addresses. They can be used to manipulate data and create complex data structures.

```c
int x = 10;
int *p = &x;  // pointer to x
*p = 20;      // change value of x to 20
printf("%d", x); // 20
```

Recursion

Recursion is a technique where a function calls itself to solve a problem.

```c
int factorial(int n) {
    if (n == 0) {
        return 1;
    } else {
        return n * factorial(n - 1);
    }
}

int result = factorial(5); // 120
```