C Reference Sheet

Getting Started:

Complete C Program:

```
                                          comment
/* A Complete C Program */

#include <stdio.h>                      directive

main()                                  function
{
   printf("\nC you later\n");             statement
}
```

Escape Sequences: \n \t \" \\

Primary Data Types:

Data Types	Examples	Conversion Specifier
int	1, 0, 7, -15	%d
float	1.0, -12.56, 3.14	%f
char	'A', 'a', '\n', '7'	%c

Declaring Variables and Constants:

```
float myGuess = 0.0;
const int DAYS = 5;
```

Printing Values:

```
printf("DAYS equals %d \n", DAYS);
```

Keyboard Input:

```
printf("Enter a number: ");
scanf("%f", &myGuess);
```

Operators (grouped by precedence):

increment, decrement	++, --
multiply, divide, modulus	*, /, %
add, subtract	+, -
relational comparisons	>, >=, <, <=
equality comparisons	==, !=
and	&&
or	||
assignment	=, +=, -=, *=, /=, %=

Conditions:

if Structures:

```
                          condition
if (floor >= 13)
{                                          executed when
   actualFloor = floor - 1;                condition is
}                                          true
else if (floor >= 0)
{                                      second condition
   actualFloor = floor;                (optional)
}
else
{
   printf("Floor negative \n");
}
                                        executed when
                                        all conditions are
                                        false (optional)
```

Conditions (continued):

Compound Conditions:

A	B	A||B	A&&B
false	false	false	false
false	true	true	false
true	false	true	false
true	true	true	true

switch Structures:

```
                             must be an
                             integral type
switch (cResponse)
{
   case 'a':
   case 'A':
      printf("You selected a or A\n");
      break;
   case 'b':
   case 'B':
      printf("You selected b or B\n");
      break;
   case 'c':
   case 'C':
      printf("You selected c or C\n");
      break;
}
```

Loops:

```
                            condition
while (balance < TARGET)
{
   year++;                                      executed when
   balance = balance * (1 + rate / 100);        condition is
}                                               true
```

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

```
do
{
   printf("Enter a positive integer: ");        executed at
   scanf("%d", &input);                         least once
}
while (input <= 0);
                         condition
```

Functions:

Function Prototypes (at beginning of the program):

function name

return type

parameter types

```
int addTwoNumbers(int, int);
```

pass by value

```
void printBalance(int);
```

pass by reference

```
int userInput(float &);

void displayMenu();
```

Sample Calls (in `main()` or another function):

```
displayMenu();
int answer;
answer = addTwoNumbers(3, 5);
```

Function Definition:

```
int addTwoNumbers(int a, int b)
{
  int sum = 0;
  sum = a + b;
  return sum;
}
```

Arrays:

Declaration and initialization:

indexes 0 through 99

```
int dollars[100];
float values[15] = {1.1, 2.2, 3.1, -1};
```

Accessing individual elements:

```
dollars[3] = 17;

for (i=0; i<15; i++)
  printf("%f ", values[i]);
```

Pointers:

Declaration and initialization:

```
int a = 14;
int b = 15;

int * iPtr;
iPtr = &a;
```

"address of" operator

```
int * anotherPtr = &b;
```

Accessing pointers and values:

```
// assign an address to another pointer
anotherPtr = iPtr;

// change the value stored in the memory
// location being pointed to
*iPtr = 3;

// print the address held be a pointer
printf("%x \n", iPtr);

// print the value being pointed to
printf("%d \n", *iPtr);
```

indirection (or dereference) operator

Strings:

C Strings are character arrays:

leave room for the NULL character

```
char fname[30];
char lname[30] = "Sawyer";
```

Input / Output:

allows entry of a string that contains spaces

```
scanf("%s", fname);
gets(lname);
printf("Hi %s %s \n", fname, lname);
```

String Functions (`#include <string.h>`):

s and s1 are C Strings, c is a char

length of s	`strlen(s)`
copy s1 to s	`strcpy(s, s1)`
concatenate s1 after s	`strcat(s, s1)`
compare s to s1	`strcmp(s, s1)`
pointer to first c in s	`strchr(s, c)`
pointer to first s1 in s	`strstr(s, s1)`

Character Functions (`#include <ctype.h>`):

c is a char

alphanumeric?	`isalnum(c)`
alphabetic?	`isalpha(c)`
decimal digit?	`isdigit(c)`
whitespace?	`isspace(c)`
convert to lower case	`tolower(c)`
convert to upper case	`toupper(c)`

Data Structures:

Declaring a `struct`:

```
typedef struct {
   int x;
   int y;
} point;
```

Declaring a variable and accessing members:

```
point first;
first.x = 1;
first.y = 4;
printf("(%d, %d) \n", first.x, first.y);
```

File Input / Output:

Declaring a FILE pointer:

```
FILE * inputFile;
FILE * outputFile;
```

Opening a file:

r for read

```
inputFile = fopen("file1.txt", "r");
outputFile = fopen("file2.txt", "w");
```

w for write
a for append

Input / Output:

```
fscanf(inputFile, "%d", &x);
fprintf(outputFile, "%f \n", 3.14);
```

Closing a file:

```
fclose(inputFile);
fclose(outputFile);
```