**Poco the AI Penguin’s C (+ Linux) Cheat Sheet**

Welcome to the C workshop we hope you learn a lot today! Some tips for C, firstly every instruction must end using ; (semicolon) in order for it to be read. Along with this all your code you will run today will use the format:

#include <stdio.h>  
  
int main(void) {  
 *// your code*  
 return 0;  
}

This will allow you to run everything we need today. To open a terminal in Visual Studio Code use **ctrl + shift + `**. Along with this we would also like to note that **//** is used to insert comments into our code that the program won’t read.

From all of us at CompClub Good Luck!

**Linux Command Line Inputs**

|  |  |
| --- | --- |
| **code *file\_name.file\_type*** | Creates a file named assigned the name and file type (in this case c) in the current directory (folder) code hello\_world.c → creates → hello\_world.c |
| **mkdir *directory\_name*** | Creates a directory (folder) in the current directory  mkdir directory\_name → creates → directory\_name |
| **ls** | Lists all current director |
| **cd *directory\_name***  **cd *..*** | Terminal enters the directory listed or in the case of .. exits out of current directory |

**Linux Command Line Compiler Inputs**

|  |  |
| --- | --- |
| **dcc *file\_name.c* -o *program*** | Compiles the C code in the file to a machine code program that can be run by the terminal |
| **./*program*** | Executes the machine code program |

**Printing**

|  |  |
| --- | --- |
| **printf(“*print this text*”);** | Prints the text contained within the quotation marks |
| **/n** | Print a new line |
| **\** | Used to remove special meaning from a character  \ on it’s own can’t be used to print \ as it looks for the value after it thus we need to double up to print one \\ prints \  Likewise if we need to print “ and we used “ to enclose our print statement we use \” to print the value |

**Variables**

|  |  |
| --- | --- |
| **int *variable\_name*;** | Designate an integer variable |
| **%d** | Command to recognise an integer variable |
| **char *variable\_name*;** | Designate a character variable |
| **%c** | Command to recognise a character variable |
| **double *variable\_name*;** | Designate a double variable (number with decimal places) |
| **%lf** | Command to recognise a double variable |

**Operators**

|  |  |
| --- | --- |
| **+** | Addition |
| **-** | Subtraction |
| **\*** | Multiplication |
| **/** | Division |

**Scanning**

|  |  |
| --- | --- |
| **scanf(“*%recognise\_command*”,&*variable*);** | Waits for an input into the terminal and when received assigns it to the variable  scanf(“%d”, &my\_num); |
| **scanf(“%*recognise\_command\_1* %*recognise\_command\_2*”, &*variable\_1*, &*variable\_2*);** | Waits for two inputs into the terminal and when received assigns it to the two variables as listed  scanf(“%d %c”, &my\_num, &my\_char); |

**Control**

|  |  |
| --- | --- |
| ***value\_1* == *value\_2*** | Condition which two values are equal |
| ***value\_1* > *value\_2*** | Condition which value 1 is greater than the value 2 |
| ***value\_1* >= *value\_2*** | Condition which value 1 is greater or equal to the value 2 |
| ***value\_1* < *value\_2*** | Condition which value 1 is less than the value 2 |
| ***value\_1* <= *value\_2*** | Condition which value 1 is less or greater to the value 2 |
| **if (*condition*) {**  ***code instructions here*;**  **}** | If *condition* is true execute the code within the braces (curly brackets) |
| **if (*condition\_1*) {**  ***first code instructions here*;**  **} else if (*condition\_2*) {**  ***second code instructions here*;**  **}** | If *condition\_1* is true execute the *first code instruction*, then ignore the else if section  If *condition\_1* is false then test *condition\_2*. If *condition\_2* is true then run the *second code instructions* |
| **if (*condition*) {**  ***first code instructions here*;**  **} else {**  ***second code instructions here*;**  **}** | If *condition\_1* is true execute the *first code instructions*, then ignore the else section  If *condition\_1* is false then run the *second code instructions* |
| **while (*condition*) {**  ***code instructions here*;**  **}** | While the *condition* is true run *code instructions* once it reaches the end test if the condition is still true, if it is run the instructions again if not exit the loop |