

Basal Ganglia & C Programming Cheat Sheet

Basal Ganglia Loops: Pathway Organization

The basal ganglia operate through distinct parallel loops, each characterized by specific cortical inputs, striatal components, pallidal outputs, thalamic targets, and ultimate cortical projections.

I. Body Movement Loop

Cortical Input: Primary motor cortex, Premotor cortex, Supplementary motor cortex

Striatum: Putamen

Pallidal Component (from Striatum): Lateral globus pallidus, internal segment

Basal Ganglia Output to Thalamus: Globus pallidus, internal segment; Substantia nigra pars reticulata

Thalamic Target: Ventral lateral and Ventral anterior nuclei

Cortical Targets: Motor cortex, Premotor cortex, Somatosensory cortex

II. Prefrontal Loop

Cortical Input: Dorsolateral prefrontal cortex

Striatum: Anterior caudate

Basal Ganglia Output to Thalamus: Globus pallidus, internal segment; Substantia nigra pars reticulata

Thalamic Target: Mediodorsal and Ventral anterior nuclei

Cortical Targets: Dorsolateral prefrontal cortex

III. Limbic Loop

Cortical Input: Amygdala, Hippocampus, Orbitofrontal cortex, Anterior cingulate cortex, Temporal cortex

Striatum: Ventral striatum (nucleus accumbens)

Pallidal Component (Basal Ganglia Output to Thalamus): Ventral pallidum

Thalamic Target: Mediodorsal nucleus

Cortical Targets: Amygdala, Hippocampus, Orbitofrontal cortex, Anterior cingulate cortex, Temporal cortex

General Pathway Components

The overarching flow common to these loops involves a series of sequential relays:

Cortical Input (e.g., Frontal cortex)

Striatum

Pallidum

Thalamus

Cortical Targets (e.g., Frontal cortex)

C Program: Overview and Key Concepts

This program prompts the user to enter a character and then checks if the input matches 'Y' (or a specific multi-character literal "Y"). It demonstrates basic C programming structures including library inclusion, the `main` function, variable declaration and usage, input/output operations, and conditional statements.

C Program Code Example

```
#include <stdio.h>

int main(void) {
    char c;
    puts("Freshman: ");
    C = getc(stdin); // Note: Original text uses uppercase 'C' for this variable assignment
    if (C == 'Y' || c == "'Y'") // Note: Original text uses 'C' then 'c', and "'Y'" as arguments
    {
        puts("Welcome\n");
    } else {
        puts("Sorry, try again.\n");
    }
    return 0;
}
```

Program Key Concepts

1. stdio.h (Standard Input/Output Library)

Description: STDO.H is the standard input/output library.

Functions Used:

- puts(): writes strings
- getc(): reads a character

2. The main Function

Execution Start: "main" is where an application starts its execution.

Return Type (int): int is the return type of the function. The main function always returns int since it must return the error code.

Arguments (void): void means that the function is not taking any arguments.

3. Variables

Declaration: Declaring variables. (e.g., char c;)

Usage: using variables. (Used to store input, in conditional checks, etc.)

Code Walkthrough / Line-by-Line Explanation

```
#include <stdio.h>
```

- This line includes the standard input/output library (`stdio.h`), which provides functions like `puts()` and `getc()` used in the program.

```
int main(void)
```

- This defines the `main` function, the entry point of the program.
- `int`: Specifies that the function returns an integer value.
- `void`: Indicates that the function takes no arguments.

```
char c;
```

- This line Declaring variables. It declares a character variable named `c`, which will be used to store a single character.

```
puts("Freshman: ");
```

- This function call uses `puts()` to display the string "Freshman: " to the console, prompting the user for input.

```
C = getc(stdin);
```

- This line reads a single character from the standard input (`stdin`) using the `getc()` function.
- The character read is then assigned to the variable `C`. (Note the original text uses an uppercase `C` here, which might be a different variable than `c` declared earlier, or a typo). This demonstrates using variables.

```
if (C == 'Y' || c == "'Y'")
```

- This is a conditional statement that checks if the character stored in `C` (from input) is 'Y' (case-sensitive) OR if the character stored in `c` (which was not assigned a value prior to this point, if `C` and `c` are distinct) is the multi-character literal "'Y'".
- The literal "'Y'" is typically not used for single character comparisons and might be a typo for 'Y'.

```
puts("Welcome\n");
```

- If the `if` condition evaluates to true, this line prints "Welcome" followed by a newline character.

```
else puts("Sorry, try again.\n");
```

- If the `if` condition evaluates to false, this line prints "Sorry, try again." followed by a newline character.

```
return 0;
```

- This line exits the `main` function and the program, returning an integer value of 0 to the operating system. A return value of 0 typically indicates successful execution.