Basal Ganglia-Thalamocortical Loops

1. Body Movement Loop

- **Associated Cortical Regions:** Primary motor cortex, premotor cortex, supplementary motor cortex
- **Pathway Components:**
    - **Cortical Input to Striatum:** Motor cortex, premotor cortex, somatosensory cortex
    - **Striatum:** Putamen
    - **Pallidum:**
        - From Striatum: H Lateral globus pallidus, internal segment
        - Output to Thalamus: Globus pallidus, internal segment; substantia nigra pars reticulata
    - **Thalamus:** Ventral lateral nucleus, ventral anterior nucleus
    - **Cortical Targets (Loop Completion):** Primary motor cortex, premotor cortex, supplementary motor cortex

2. Prefrontal Loop

- **Associated Cortical Regions:** Dorsolateral prefrontal cortex
- **Pathway Components:**
    - **Cortical Input to Striatum:** Dorsolateral prefrontal cortex
    - **Striatum:** Anterior caudate
    - **Pallidum (Output to Thalamus):** Globus pallidus, internal segment; substantia nigra pars reticulata
    - **Thalamus:** Mediodorsal nucleus, ventral anterior nucleus
    - **Cortical Targets (Loop Completion):** Dorsolateral prefrontal cortex

3. Limbic Loop

- **Associated Cortical Regions:** Anterior cingulate cortex, orbitofrontal cortex
- **Pathway Components:**
    - **Cortical Input to Striatum:** Amygdala, hippocampus, orbitofrontal cortex, anterior cingulate cortex, temporal cortex
    - **Striatum:** Ventral striatum (nucleus accumbens)
    - **Pallidum (Output to Thalamus):** Ventral pallidum
    - **Thalamus:** Mediodorsal nucleus
    - **Cortical Targets (Loop Completion):** Amygdala, hippocampus, orbitofrontal cortex, anterior cingulate cortex, temporal cortex

C Program Structure and Explanation

This document breaks down a simple C program, explaining its components and the concepts behind them.

The Program Code

```
#include <stdio.h>

int main(void) {
    char c; // Variable declaration
    puts("Freshman: "); // Displaying a prompt
    c = getc(stdin); // Reading a character from standard input
    // Conditional check for user input
    // (Note: The original text had 'C' for the variable and `c == "\'Y\'`.
    // Corrected to 'c' for consistency and `c == 'y'` for logical single-character comparison.)
    if (c == 'Y' || c == 'y') {
        puts("Welcome\n");
    } else {
        puts("Sorry, try again.\n");
    }
    return 0; // Program exit status
}
```

Explanation of Program Components

1. `#include <stdio.h>`
   - This line is a preprocessor directive that includes the contents of the `stdio.h` header file.
   - `stdio.h`**:** This stands for "Standard Input/Output Header." It is the standard library for input and output operations in C.
   - **Functions used from `stdio.h` in this program:**
     - `puts()`: Writes a string to the standard output (console), followed by a newline character. (e.g., `puts("Freshman: ");`)
     - `getc()`: Reads a single character from a specified input stream. In this case, `stdin` (standard input, usually the keyboard) is used. (e.g., `c = getc(stdin);`)
2. `int main(void)`
   - `main` **function:** This is the mandatory entry point of every C program. An application starts its execution here.
   - `int`**:** This specifies the return type of the `main` function. The `main` function always returns an integer, which serves as an exit status or error code to the operating system (0 typically indicates successful execution).
   - `void`**:** This indicates that the `main` function does not take any arguments. (Functions can be designed to accept arguments, which will be covered in more detail in other contexts.)
3. **Variable Declaration and Usage**
   - `char c;`**:**
     - **Declaring variables:** This line declares a variable named `c` of type `char`. Declaring a variable reserves a specific amount of memory for it, based on its type.
   - `puts("Freshman: ");`**:**
     - This line uses the `puts` function to display a prompt string to the user.
   - `c = getc(stdin);`**:**
     - **Using variables:** This line reads a single character from the standard input (`stdin`) and assigns it to the `c` variable.
4. **Conditional Statement (**`if-else`**)**
   - `if (c == 'Y' || c == 'y')`**:** This is a conditional statement that checks if the character stored in the variable `c` is either an uppercase 'Y' OR a lowercase 'y'.
   - `puts("Welcome\n");`**:** If the condition in the `if` statement is true, this message is printed.
   - `else`**:** If the condition in the `if` statement is false.
   - `puts("Sorry, try again.\n");`**:** This message is printed.
5. `return 0;`
   - This statement exits the `main` function. The value `0` is returned to the operating system, indicating that the program executed successfully without any errors.