Gamescrafters Documentation
TEXT PARSING

David Chan

Version History

2006.12.13 - Version 1.0 - First version created.

Credits

Max Delgadillo - Structure based on his generic_hash_api.tex file

Contents

1 Overview

The goal of the text parsing API is to provide a standardized way for programmers to read from stdin. It also aims to provide a simple way to integrate support for new data types as Gamescrafters grows. It is not a replacement for good programming practices. The error checking it does is minimal. It is expected that the programmer knows what they want to read from stdin and that the API will return this result.

2 Goals and Non-Goals

The documentation will go over the basics of the API and how to implement new functioniality. There will be little code behind the API revealed. The reader is assumed to know how to do memory management in C if added functionality or the string functions are called.

3 Text Parsing API

Note that stdin is flushed after any of the below function calls.

3.1 `char GetMyChar()`

This function reads in the next character from stdin and returns it.

3.2 `int GetMyInt()`

This function reads in the next integer from stdin and returns it.

3.3 `void GetMyStr(STRING str, int len)`

This function reads in a string from stdin. It will read at most len - 1 characters, with the last character reserved for `\0`. There is no error checking on the actual size of allocated memory. It is up to the programmer to ensure that len is <= the malloc'd size of memory.

3.4 `void GetMyHelper(char *format, GENERIC_PTR target)`

Calls `GetMy` with format, target, default length and no to keeping spaces. The programmer should not be concerned with this function. The only time it should be used is when adding functionality to the API.

3.5 `void GetMy(char *format, GENERIC_PTR target, int length, BOOLEAN keepSpaces)`

This function show never be called directly, instaed functionality should be added by making wrapper functions. format is the format string that will be passed to sscanf. target is a pointer to the the memory address of where to store the result from sscanf. length - 1 is the maximum amount of characters to read from stdin. keepSpaces is used to determine where to include white spaces in the output. This should only be `true` when target is a pointer to a continguous block of memory. The results are undefined if target is a pointer to a noncontiguous block of memory. After reading from stdin, the stdin buffer will be flushed. Thus if multiple nonstring entries need to be read from stdin on one line, it is advised to use `GetMyStr()` then parse the resulting string.

4 Adding Functionality

The text parsing functions are designed to easily accomodate added functionality. This functionality can be achieved through the use of wrapper functions.
Adding a function to read in positions

```
POSITION getPosition() {
  POSITION temp;
  GetMy("%llu", &temp);
  return temp;
}
```

General format of a new function

return-type function-name(*optional arguments*) {
    return-type var;

    GetMy(*format*, var);

    return var;
}