Usage examples and an overview of the library may be found at:
https://github.com/assafmuller/Bit-Bucket/raw/master/README.pdf

BitBucket Class Reference

Public Member Functions

	BitBucket ()
	BitBucket (std::string filePath)
void	**serialize** (std::string filePath)
bool	**isSet** (std::string key)
void	**print** ()
void	**printBlank** ()
void	**print** (std::function< bool(std::string key, **Bit** value)> predicate, std::ostream &out=std::cout)

Detailed Description

BitBucket is a hash map, where the keys are strings and the values are Bits. Specifically, **BitBucket** inherits from std::unordered_map, so you may use it as an STL container.

Constructor & Destructor Documentation

BitBucket::BitBucket ( )

Default constructor.

BitBucket::BitBucket (std::string filePath)

Initialize the **Bit** Bucket from a text file. The format is: <type> <name>

<type> <name>

...

For example:

int x 5

float y 3.14

string s Hello world!

auto b false

When using "auto" as a type the type of the variable will be deduced according to its value, much like the "auto" keyword in C++11.

Parameters

filePath	- Path to the text file holding the data

Member Function Documentation

bool BitBucket::isSet (std::string *key*)

Is there something in bucket[key]?

void BitBucket::print ( )

Print the entire bucket in the following format:

<type> <name> <value>

void BitBucket::print (std::function< bool(std::string key, Bit value)> *predicate*, std::ostream & *out* = `std::cout`)

Prints all cells that return true to the predicate/lambda passed in, to the output stream specified. std::cout is the default value.

Parameters

predicate	- An std::function that accepts a key and value and returns if the key/value pair should be printed
out	- An output stream

void BitBucket::printBlank ( )

Print all blank cells in the bucket. Blank cells can be created like so: int x = bucket["nonExistantCell"];

void BitBucket::serialize (std::string *filePath*)

Write the bucket's contents to a text file of the same format as described in **BitBucket** (std::string filePath).

Parameters

filePath	- Path to the text file to write to

Bit Class Reference

`#include <Bit.h>`

Public Member Functions

	Bit ()
	Bit (Variant variant)
	Bit (const char *text)
	Bit (std::string **type**, std::string value)
std::string	**type** ()
template<class T >	
	Bit (T t)
template<class T >	
	operator T ()
	operator std::string ()
Bit	**operator=** (const **Bit** &other)

Friends

std::ostream &	**operator<<** (std::ostream &out, const **Bit** &bit)

Detailed Description

The **Bit** class is a variant, able to hold bool, char, int, float or string at any given time.

Constructor & Destructor Documentation

Bit::Bit ( )

Default constructor. Sets the **Bit** to blank.

Bit::Bit (Variant *variant*)

Copy constructor.

Bit::Bit (const char * *text*)

Required so that you may initialize **Bit** from a string literal. The templated version gives a compile error when initializing from a string literal.

Bit::Bit (std::string type, std::string value)

Used when initializing from a text file or another exterior resource.

Parameters

type	- Can be bool, char, int, float, string or auto
value	- The value this variant will initially hold

template<class T >

Bit::Bit (T t) inline

Constructor used when initializing from an intrinsic variable. IE: **Bit** bit = 5;

Member Function Documentation

Bit::operator std::string ( ) inline

Specialized casting function, when converting the variant to a string. If the variant is a string, just return it. Otherwise, try to cast it to a string. If unsuccessful, returns an empty string.

template<class T >

Bit::operator T ( ) inline

Conversion to all intrinsic types but string, which has a specialization. If the variant is not a string, then it simply returns the value of the variant, in its current type. I then let a possible implicit conversion occur - This is the desired behavior. If the variant is a string, and (as mentioned before) we're converting to a type that's not a string, then try to lexically cast the string to the requested type. If unsuccessful, will return a default-initialized of the requested type.

std::string Bit::type ( )

Returns the current type of the variant in a textual format ("int", "float", ...)

Friends And Related Function Documentation

std::ostream& operator<< (std::ostream & out, const Bit & bit) friend

Allows the **Bit** class to be streamed