The StorageManager class - easy data storing

SkyProgger, 26 Oct 2014 CPOL Rate this:

★★★★★ 5.00 (1 vote)

This article describes the StorageManager class I have developed. It's created to save data beyond the Programs lifespan, as easy as possible.

Download StorageManager.zip - 360.5 KB

Introduction

The `StorageManager` class was created to allow programmers to easy store Data. This is done by simulating a new area of memory (beneath datasegment, stack and heap). The Storage. The StorageManager provides some methods to access this area. Memory allocated within this area will be stored and reloaded when closing and restarting the program. The aim was to keep it as simple as possible for the users of this class, but also to allow some more specialized methods.

The Project itself includes a full HTML-documentation, created with Doxygen (http://www.stack.nl/~dimitri/doxygen/). So this article will only feature the basic functionality of the `StorageManager`. The files itself are attatched as zipped archive.

Using the code

The `StorageManager` is a static class, which is located in the datasegment. So as soon as you include the .cpp file to your project, a StorageManager will be created. The Manager will Auto-Setup itself on Program start, loading all the data you stored before, and will auto-save it and clean itself up when the Program ends. But you can modify this behavior to customize the Manager.

The most common functions to use would be allocating and deallocating Storage. This can be done in various ways, depending on what you need, but the easiest (and most common) way is the following:

Hide Copy Code

```
#include "StorageManager.h"
#include <iostream>

int main()
{
    std::cout << StorageManager::getLoadProtocol() << std::endl;

    int *ptr = StorageManager::getVariable<int>("Int-Variable", true, 11);
    std::cout << *ptr;
    (*ptr)++;

    return 0;
}
```

The `getLoadProtocol` function will simply print out a protocol, the Manager creates on loading a file. There you can find all Variables that have been loaded and, if occured, any loading errors. (Notice: The Manager called the load function on it's construction, so at this point, the protocol already exists)

But what is more interesting here is the `getVariable` function. In short, it will do the following: It first checks the Manager, if a variable with the name "Int-Variable" already exists. If yes, it will load it and return the pointer to it. If no, it will create a new one, and forward the 11 to the variables constructor. So in this case, the integer will be set to 11. The second parameter specifies how the Manager behaves, if a variable with the given name is found, but it's not an integer or an array. I set it to true, which means, the Manager will discard the old value in this case and replace it. The outcom of this program would be 11. But if you run it a second time, it will be 12 instead, then comes 13, ... As you can see: I created a Variable on the Storage, that lives beyond the programs lifespan. and each time I run the program, i increase it's value by 1. Thats basically it, thats the StorageManagers main functionality.

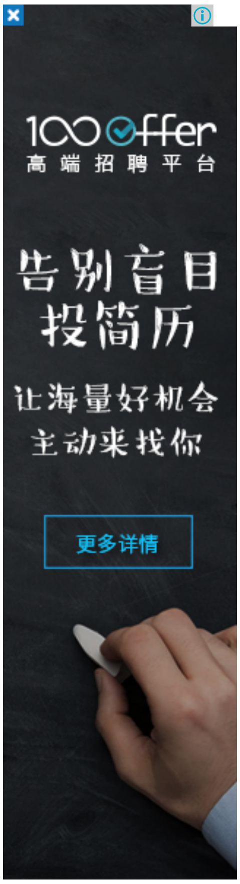
Please Notice: I did NOT delete the Memory I allocated. this is not a mistake. The Manager itself will care about Memory, and internally free it as soon as it's destructed. If you free the Pointer, this will remove it from the Storage, which means, it will be permanently deleted. It won't be stored anymore. To do this, call `removeVariable`, as done in the following example:

Hide Copy Code

```
#include "StorageManager.h"
#include <iostream>

int main()
{
    std::cout << StorageManager::getLoadProtocol() << std::endl;

    int *ptr = StorageManager::getVariable<int>("Int-Variable", true, 11);
```



```
    std::cout << *ptr << std::endl;
    (*ptr)++;

    std::cout << "delete the Int-Variable? (enter y to delete): ";
    if((char)cin.get() == 'y')
        StorageManager::removeVariable("Int-Variable");

    return 0;
}
```

In this case, if you enter 'y', the Variable will deleted. Please notice, that this will cause `ptr` to get invalid, because it will be deleted. The pointers address will be added to the `CorruptionList`. Please check the full documentation, if you want to know more about this topic.

Of course, the Manager can do much more. The most remarkable would be:

- Storing arrays (They need a own method, `getArray`)
- Easy methods to copy or move pointers between the Storage and e.g. the Heap
- The `StorageVariable` class, which allows to store pointers (with some restrictions)
- Encryption of data stored. This feature needs the CryptoPP library(http://www.cryptopp.com/). On default, the parts using the the Library are excluded with preprocessor switches

History

- 25.10.2014: First Version of the StorageManager published, Vers 1.0

License

Share

EMAIL TWITTER

About the Author

SkyProgger

Germany

No Biography provided

You may also be interested in...

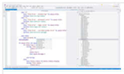
Non-Tech Factors to Consider When Choosing Your Tech Stack

How-To Intel IoT Code Samples: Range Finder Scanner

How-To Intel® IoT Technology Code Samples: Doorbell in C++

Getting Started with React

A Beginners Guide to WebGL



10 Ways to Boost COBOL Application Development

Comments and Discussions

You must Sign In to use this message board.

Search Comments Go

□ Profile popups Spacing Relaxed Layout Normal Per page 25 Update

First Prev Next

Great work	**Andres Cassagnes**	**27-Oct-14 9:01**
very useful :-) thanks for sharing !	**Matth Moestl**	**27-Oct-14 0:41**
Re: very useful :-) thanks for sharing !	SkyProgger	27-Oct-14 3:35

Last Visit: 31-Dec-99 18:00 Last Update: 8-Jun-16 2:17 Refresh 1

General News Suggestion Question Bug Answer Joke Praise Rant Admin

Use Ctrl+Left/Right to switch messages, Ctrl+Up/Down to switch threads, Ctrl+Shift+Left/Right to switch pages.

Permalink | Advertise | Privacy | Terms of Use | Mobile
Web01 | 2.8.160531.1 | Last Updated 26 Oct 2014

Layout: fixed | fluid