Appeal of Memory Management in Object Oriented Programming Language C++

Sinduja Reddy Anantha
Department of Computer Science
University Of Bridgeport
Bridgeport, USA
sanantha@my.bridgeport.edu

***Abstract*—As a general rule Dynamic Memory Management (DMM) is an expensive segment in numerous programming frameworks. Related to memory C projects do expend around 30% of the project runtime in memory allotment and de-allocation. C++ as a rule perform making and erasing an item. According to old investigation C++ projects do have more than 10 times memory allotment and de-portion than C-dialect programs. This paper portrays from where dynamic memory distributions come and how dynamic memory summons do happen in C++ programs. Initially, this paper spells out the speculation of cases which invokes the element memory administration explicitly and additionally verifiably. Secondly, to do examination on the theory source code level following tracing tool is accounted for strategy. Eventually, outcomes include behavioral examples of memory portions. By spending these examples, most likely we scale up the reusability of the assets.**

Keywords— Dynamic Memory Management (DMM), Object Oriented Programming (OOP) & Software tools.

I. INTRODUCTION

Object arranged frameworks are intense through the years. The growing up utilization in picking C++ in the business is unmistakably a clear that more applications would be created utilizing C++ dialect. To gage the execution issues for occurrence the conduct of its element memory distribution and de-allocation, I have to do investigative investigation. This kind of examination results are utilized as a part of 3 separate ways.

First principal, a developer's rule and proposition is emphatically dependent on his examination results. So we do know why the C++ projects are had a tendency to be abate. Secondly, design to backup instruction set structural planning or coprocessor, which is produced dependent on the exploration results. For example, equipment helped memory administration is the most ideal approach to encourage the issue of being memory concentrated.

Finally, to recognize memory spillage programming instruments are produced taking into account the examination results. Additionally, execution of element memory administration (DMM) is scaled up, if the distribution examples are

This paper exhibits another way to examine memory allotment conduct at level of source code. We begin with a grouping of every conceivable circumstance that could summon element memory administration (DMM) in C++.

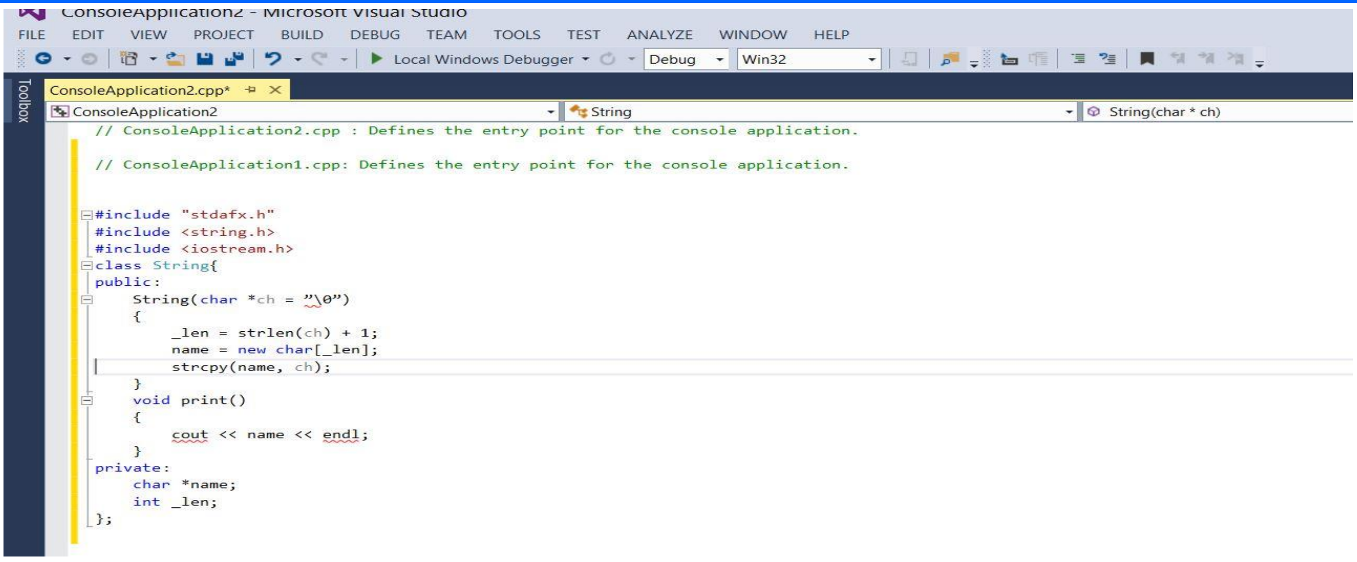
In my speculations, this memory allotment examples are fitting to either an application level or C++ vernacular. Case in point, DMM summons from constructors, copy constructors or errand manager = over-troubling are exceptional to question organized programming with C++. Application specific part lives up to expectations which summons new or eradicate head unequivocally are application suitable. To research my speculations on the bit direct, a "Take after instrument" has been created to implement C++ source programs. This gadget does take C++ source code as an information and do inserts C++ decrees into the conjectured segment outlines. Exactly when the implemented C++ undertakings are executed in meantime the insights of component memory appropriation (DMA) would be delivered. These delivered takes after then are poor around a substitute gadgets to get the estimations. The C++ programs we followed in this trial are open source programming devices. These devices do contain Java compiler, debugger, CORBA and Venus. Underneath zone 2 spells out the circumstances how the DMM is summoned in C++ tongue programs.

II. DIFFERENT SCENARIOS IN DYNAMIC MEMORY ALLOCATION (DMA) IN C++

Mainly the element memory portion (DMA) in C++ is summoned by new administrator. The new administrator is actualized utilizing malloc() capacity. In this section, I do illustrate 5 sorts of cases they are:

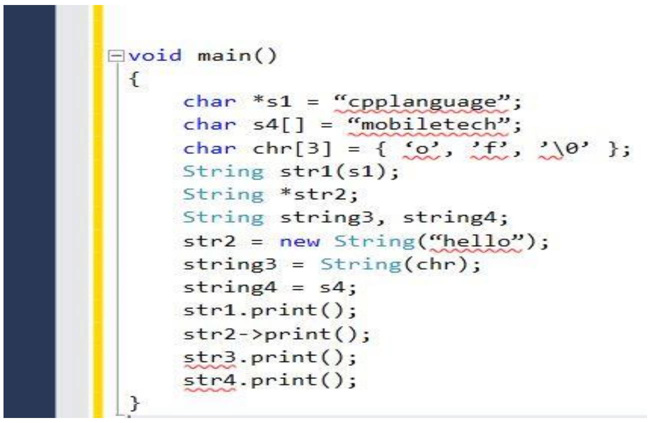
A) Constructors

A constructor conjures new administrators either expressly or verifiably. A constructor which utilizes new administrator is summoned either unequivocally or certainly. Do figure a String class is formed as taking after


ConsoleApplication2 - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ANALYZE WINDOW HELP
Local Windows Debugger
Debug
Win32
Toolbox
ConsoleApplication2.cpp*
ConsoleApplication2
String
String(char * ch)

```
// ConsoleApplication2.cpp : Defines the entry point for the console application.

// ConsoleApplication1.cpp: Defines the entry point for the console application.


#include "stdafx.h"
#include <string.h>
#include <iostream.h>
class String{
public:
    String(char *ch = "\0")
    {
        _len = strlen(ch) + 1;
        name = new char[_len];
        strcpy(name, ch);
    }
    void print()
    {
        cout << name << endl;
    }
private:
    char *name;
    int _len;
};
```

In the example mentioned, one constructor is proclaimed. This constructor is conjured certainly in the item creation and introduction. Primary capacity in the system have the underneath code to hotshot the conjuring of the constructor.



```
void main()
{
    char *s1 = "cpplanguage";
    char s4[] = "mobiletech";
    char chr[3] = { 'o', 'f', '\0' };
    String str1(s1);
    String *str2;
    String string3, string4;
    str2 = new String("hello");
    string3 = String(chr);
    string4 = s4;
    str1.print();
    str2->print();
    str3.print();
    str4.print();
}
```

In the above code we do use new word to make and introduce a String article. The object conjures the String (char*) constructor. In the above code, there are 3 things are going on

- To make a class object, we call a constructor.
- Memory wise assessment from recently made item to the article string 3 and
- Disposing the article we created.

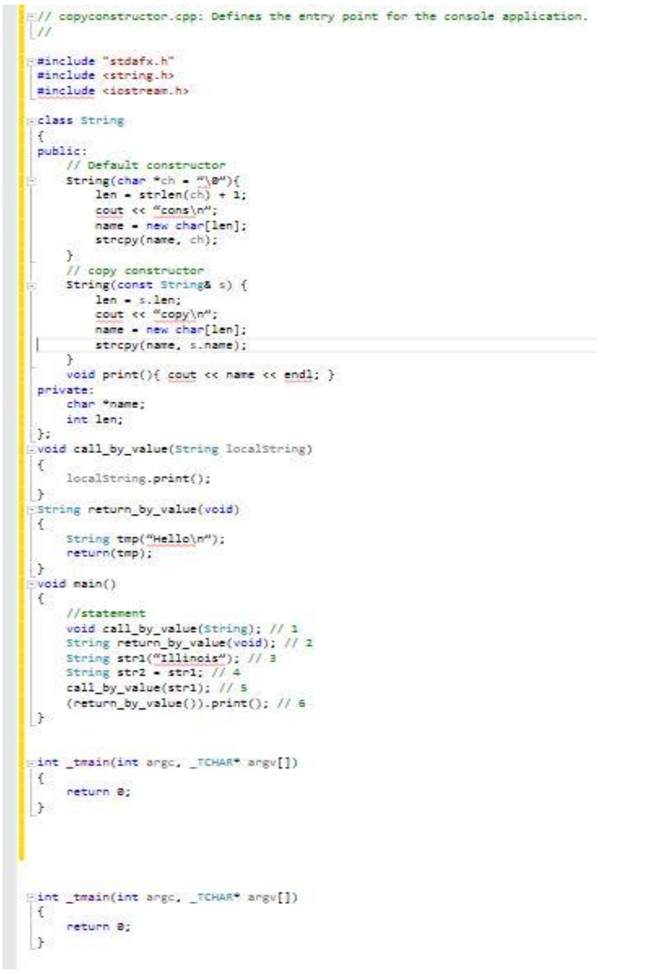
The information individual from the impermanent article does not summon dynamic memory portion (DMA), in lieu of the information part is hidden away on the stack as a compiler created programmed variable. The constructor in (a) may conjure dynamic memory designation (DMA). At whatever point the constructor is summoned the administrator new is conjured off. This is the way, the item is made off alertly in the stack memory.

B) Copy Constructor

The duplicate constructor meets expectations as opposed to the constructor and is conjured certainly just. The default duplicate constructor does perform part astute duplicate. The duplicate constructor, as opposed to the constructor, can summon new administrators verifiably just. The default duplicate constructor does duplicate just the pointer yet not a whole string. A client characterized duplicate constructor must be characterized before a profound duplicate is performed. Truth be told, a profound duplicate constructor does conjure dynamic memory distribution. The duplicate constructor most likely be summoned in the accompanying ways:

- Non-static member functions.
- They should not contain parameters or a declared return types.

Function should be returned.

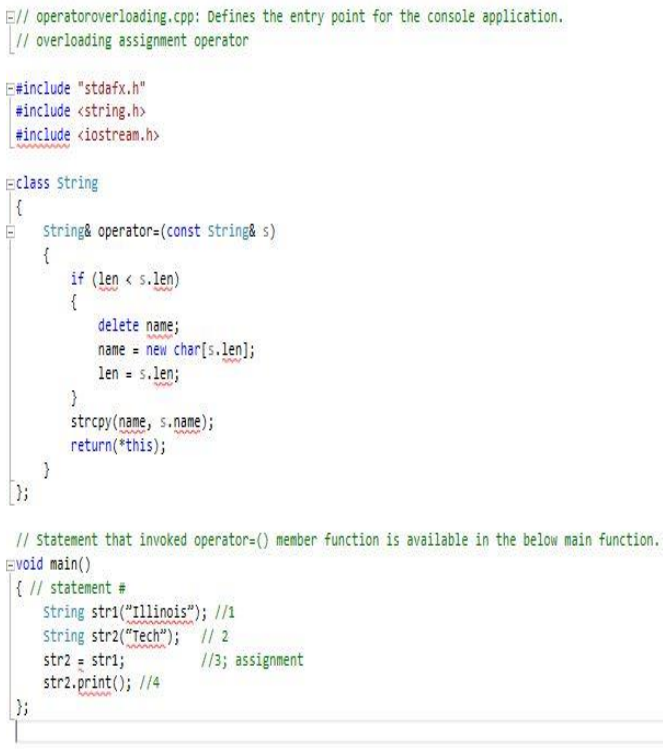


```
// copyconstructor.cpp: Defines the entry point for the console application.
//

#include "stdafx.h"
#include <string.h>
#include <iostream.h>

class String
{
public:
    // Default constructor
    String(char *ch = "\0"){
        len = strlen(ch) + 1;
        cout << "cons\n";
        name = new char[len];
        strcpy(name, ch);
    }
    // copy constructor
    String(const String& s) {
        len = s.len;
        cout << "copy\n";
        name = new char[len];
        strcpy(name, s.name);
    }
    void print(){ cout << name << endl; }
private:
    char *name;
    int len;
};
void call_by_value(String localString)
{
    localString.print();
}
String return_by_value(void)
{
    String tmp("Hello\n");
    return(tmp);
}
void main()
{
    //statement
    void call_by_value(String); // 1
    String return_by_value(void); // 2
    String str1("Illinois"); // 3
    String str2 = str1; // 4
    call_by_value(str1); // 5
    (return_by_value()).print(); // 6
}


int _tmain(int argc, _TCHAR* argv[])
{
    return 0;
}



int _tmain(int argc, _TCHAR* argv[])
{
    return 0;
}
```

The call-by-value () method is used by the copy constructor which is invoked as shown in the above program [statement 5].The return- by-value () method is also used to invoke the copy constructor in [statement 6].

C) Operator = Overloading Assignment

Default task administrator does perform part savvy task. To perform a "Profound duplicate" like task between class items, over-burdening the task administrator is an absolute necessity. To over-burden the task operator= which actualizes a profound duplicate like task among class, articles, dynamic memory management summoned. The part work administrator = () that does perform a profound duplicate presumably conjure dynamic memory allotment (DMA). In the String class said in the recent past, a task administrator is over-burden utilizing the beneath part work.



```
// operatoroverloading.cpp: Defines the entry point for the console application.
// overloading assignment operator

#include "stdafx.h"
#include <string.h>
#include <iostream.h>

class String
{
    String& operator=(const String& s)
    {
        if (len < s.len)
        {
            delete name;
            name = new char[s.len];
            len = s.len;
        }
        strcpy(name, s.name);
        return(*this);
    }
};

// Statement that invoked operator=() member function is available in the below main function.
void main()
{ // statement #
    String str1("Illinois"); //1
    String str2("Tech");   // 2
    str2 = str1;           //3; assignment
    str2.print(); //4
};
```

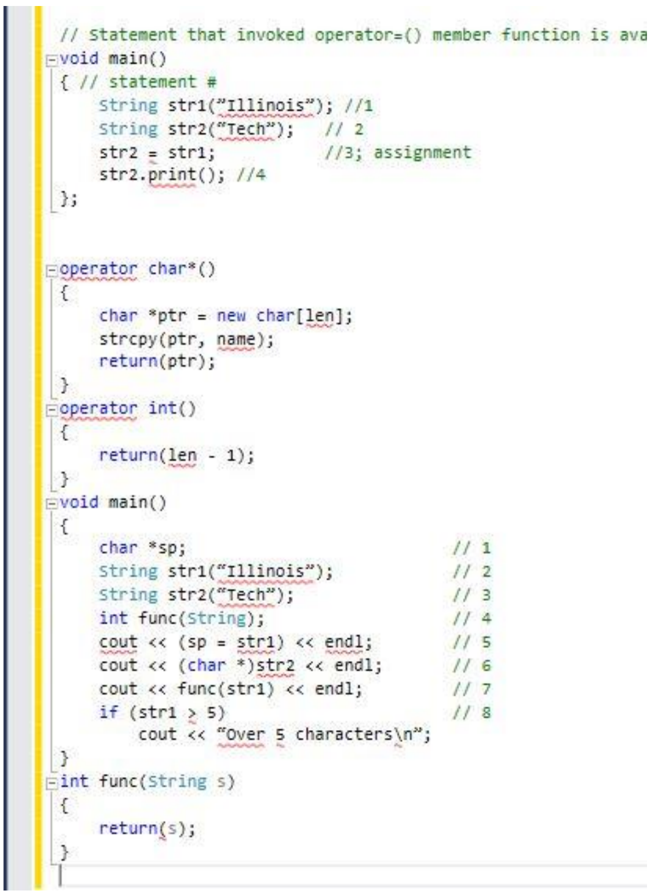
In the above code proclamation three does summon client characterized task administrator. At that point dynamic memory portion may be summoned. In this was this is stretched out to some other capacities all inclusive or mainly, which is utilized to over-burden C++ administrators.

D) Type Conversions

C++ does encourage a procedure to change over client characterized date sort to an implicit sort information sorts. Sort transformation is utilized as a part of C++ to change over a client characterized sort to an implicit sort. Such transformation might likewise summon new administrators. This kind of client characterized information sort transformation is executed utilizing an uncommon part work with a type of administrator sort() { . . . }.Some of normal tenets about these capacities are:

- Non-static member functions.
- They should not contain parameters or return declare types.
- Function should be returned.

All sorts of transformations do happen verifiably in task articulation, social outflow, and parameters to capacities and qualities came back from capacities. Having portrayed beneath, how about we include two such kind of transformation capacities to String class.

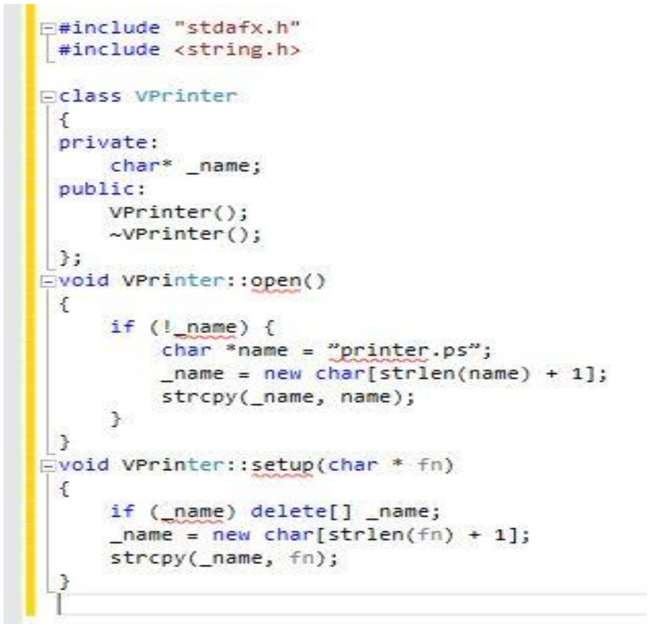


```
// Statement that invoked operator=() member function is ava
void main()
{ // statement #
    String str1("Illinois"); //1
    String str2("Tech");   // 2
    str2 = str1;           //3; assignment
    str2.print(); //4
};


operator char*()
{
    char *ptr = new char[len];
    strcpy(ptr, name);
    return(ptr);
}
operator int()
{
    return(len - 1);
}
void main()
{
    char *sp;                               // 1
    String str1("Illinois");                // 2
    String str2("Tech");                    // 3
    int func(String);                       // 4
    cout << (sp = str1) << endl;            // 5
    cout << (char *)str2 << endl;           // 6
    cout << func(str1) << endl;             // 7
    if (str1 > 5)                           // 8
        cout << "Over 5 characters\n";
}
int func(String s)
{
    return(s);
}
```

The announcement 5 and 6 do summon change, administrator char*(), verifiably and expressly. Proclamation 7 does summon change, administrator int() through returned worth from capacity call. Explanation 8does conjure transformation, administrator int(), verifiably.

E) Member functions (User Defined Functions)

Predominantly, the way that a part capacity does summon DMA is never characterized effortlessly. Everything depends on the utilization. These are implemented by programmer as per his requirement. They give diverse administration to the class object.



```
#include "stdafx.h"
#include <string.h>

class vPrinter
{
private:
    char* _name;
public:
    vPrinter();
    ~vPrinter();
};
void vPrinter::open()
{
    if (!_name) {
        char *name = "printer.ps";
        _name = new char[strlen(name) + 1];
        strcpy(_name, name);
    }
}
void vPrinter::setup(char * fn)
{
    if (_name) delete[] _name;
    _name = new char[strlen(fn) + 1];
    strcpy(_name, fn);
}
```

A part capacities, those install and opening, conjuring new administrator unequivocally. In separation its anything but difficult to see that such DMA summons are truly fluctuate from the initial 4 cases clarified previously. The fifth case needs to do with application just. Which is the reason, we disengage the fifth one from whatever remains of initial 4 cases.

III. TRACING TOOL

An apparatus does take source code of C++ as data. C++ proclamations as imprints into the guessed portion designs. This apparatus is actualized utilizing C++ code and additions starting and completion marks for every element memory allocation situation. Every imprint embodies data about part capacity. So the class' part capacities are followed as they are summoned. An execution of DMM in C++5 is additionally changed to the way that every summon to the DMM does produce follows.

At the point when the implemented C++ projects are executed and assignment are produced. These follows then are gaged by an alternate device to bring the insights. The summon recurrence and size of the portion. Case in point these are gathered utilizing the analyzer. The itemized information is clarified in the following segment. The C++ projects followed in this investigation are open source and are accessible programming applications. These projects incorporate Guavac, CORBA consistent. They speak to a few separate applications.

IV.RESULTS:

I do figure out that a considerable lot of these C++ projects did not use article situated programming system. So these projects are similar to C program than a C++ program. As a rule, just 1 or 2 classes are characterized and instantiated in whole program. These C++ projects are rejected from my investigation. Omni Broker gives the source code to non-business employments. These projects speak to late changes in OOP and are coded by expert designers.

Over 4 noteworthy projects that are utilized as a part of this investigation, everyone is considered dreadfully in the examination. C++ applications are picked and explored sooner rather than later. For Guavac which has no screen connection, 5 speculated types are gathered and displayed for whole session. The projects that have screen co-operations information are gathered and displayed for every cooperation session. This aids us to watch and screen the designation design as the application is running.

V. CONCLUSION

This paper does show a study about element memory distribution conduct in C++. The level of following device for Source Code is produced to explore the theory. C++ programs, which are followed incorporate Java compiler, CORBA agreeable structures. This projects are most current improvements in Object Oriented Programming and speak to no screen applications to GUI. From all these perceptions, duplicate constructors mostly summon DMA. It is useless that duplicate constructors are conjured certainly in C++ programs. It implies new kid on the block software engineers may not mindful of the conjuring of duplicate constructor as they code the C++. Duplicate constructors are conjured in a few courses, for example, passing a class item to a capacity utilizing call-by-worth, giving back a class object from a capacity utilizing return by-quality and actually instating a class object. Having a thought of the expense of element memory administration (DMM) and the programming styles that could conjure DMA is essential in a productive C++ program. The aggregate rate of DMA conjuring in view of the initial 4 conjectured cases get shift from application to application going from 27% to 100%.There is a reliable designation design for a particular application paying little mind to its include. With a reliable DMA conjuring example, a profile based memory administration system is conceivable and is executed effectively.

REFERENCES

1.Adams, “Windows Visualization Programming with c/c++”, and Barton, Department of Computer Science, Univ. of Colorado, Boulder, CO, January1995.

2. Anderson, “C++ Programming and Fundamental Concepts” and E. F. Gehringer, “A High-Performance Memory Allocator for Object-Oriented Systems,” March, 1996. pp. 357-366.

3. Buzzi-Ferraris Callan Titled as “Addison Comp”, “Scientific C++ Building Object-Oriented Systems, Applications in C++”.