Initialization & Finalization

CSCI 305
John Nelson
Matthew Gallagher

Introduction

All programs require variables to be able to accomplish the tasks laid out by the developer. Additionally, if the language is object oriented it will need objects, which are a data type that contains parameters set by the developer. Objects and variables are utilized in a program through the use of initialization and finalization. Initialization is the assignment of space in memory and the assignment of values to objects and variables. Finalization is the exact opposite where a variable is removed from memory when it is no longer used. In this paper we will be taking a brief look at the overview and purpose of these processes; the history of their use in languages; a sample of the languages in which they are used; the challenges associated with their utilization; the pros, cons and why we chose to explore initialization and finalization.

Why we chose Initialization & Finalization

Over the course of our studies at Montana Tech we have spent the majority of our time learning to program using object-oriented languages. Whether it be our first year learning Java or our second year learning C++, every time we created a new program we were required to create a constructor for the class or initialize our variables. This made us think about what was the programming function behind this and in turn lead us to the concept of initialization. We also realized that if an object or variable is initialized it must be removed from memory at the end of its lifespan. Realizing that initialization and finalization are a major part of the languages we studied at Montana Tech we decided to learn more about initialization and finalization. At the beginning of our discovery about these processes, the first information we learned about this topic was the overview and purpose of initialization and finalization.

Overview/Purpose

Initialization and finalization are a major construct of most object-oriented languages. When an object/variable is created it goes through the process of initialization, the purpose is to facilitate the assignment of space in memory and values to those objects and variables. Initialization of objects is accomplished by constructors and a class depending on the language may have one or more. A constructor's purpose in the initialization is to either initialize a default object in memory like Test() or initialize an object that has parameters assigned to it such as Test(String Qs, int Pts). Initialization of variables is accomplished by assigning it a value such as int = 1 or String = “Hello”. Especially important when an object/variable is initialized is considering whether to pass the values by reference or by value. To pass by value, memory receives a copy of the object or variable and to pass by reference, memory receives the actual instance of the object or variable.

On the opposite side when an object/variable is at the end of its life span and no longer needed by a program it undergoes the process of finalization. Finalization occurs by the use of either garbage collection or a destructor. Garbage collection works exactly as it sounds where once an object/variable is no longer used a garbage collector deallocates it from memory to free up space. The destructor works almost in the same manner, with the major difference being it is called manually. This fully ensures an object/variable is deallocated from memory and is especially important when a program uses resource intensive objects or variables. Overall, the purpose of initialization and finalization is to allow for the creation and destruction of objects and variables being used in a program. The idea of initialization and finalization stretches all the way back to the 1950s with early versions of Fortran and Lisp. In the next section we will take a look at the history or how initialization and finalization came to be used in programming.

History

Initialization has been a part of computer programming languages since the beginning to initiate variables. While there are two types of programming languages: function and object oriented, we will only be discussing the initialization as it pertains to what was mentioned in the textbook. The first object-oriented programming language is Simula 67 created in 1967. In this version of Simula it introduced objects, classes, inheritance, subclasses and constructors. The use of initialization allowed developers to create an object at runtime and assign values to a space in memory for that object. A language that adopted the class concept from Simula was Smalltalk. A project that was aimed at creating a dynamic OOL (Object Oriented Language). This allowed class objects and methods to be interpreted as metadata at runtime.

The most successful OOL arguably is C++. Speaking from experience, this is a language that everyone will take in computer science at Montana Tech. C++ was created in 1983 by Bjarne Stroustrup as an extension of the language C. C++ is a lower level programming language compared to newer languages such as Python, Ruby, and Java. Python was released in 1991 and is a OOL used to help programmers clear logical code for projects. Python is a very popular programming language as of the past decade. Java was introduced in 1995, the same year as Ruby. Both languages are high-level OOL Influenced by SmallTalk and C++.

When clearing the memory that is holding the object after not being referenced anymore, the process of finalization occurs. The first use of finalization was garbage collection in Common Lisp created in 1958. Garbage collection is the process of removing an object or variable from memory that is no longer referenced in the source code. All languages have their own way of dealing with the “garbage”. Java has an automatic garbage collection system that automatically clears memory where an object that was initialized is no longer referenced. In C++, the way to dispose of garbage is by using a destructor. The destructor will delete the data held in the memory location of the object. Coding in C++ requires efficient memory allocation and usage. Fortunately, C++ has the construct of smart pointers which has assisted programmers to collect garbage and dispose of it. While fascinating, the historical aspect of initialization and finalization also provides the basis for the usage of these processes in the various languages used by programmers.

Languages Utilization

In this section we will be taking a look at some of the different languages that use initialization and finalization. Before beginning, it is important to understand that all of the languages utilizing this feature are either an object-oriented language or share a feature with object-oriented languages. The languages include Java, Fortran 2003, Common Lisp, Simula, and Python and C++. In all of the languages mentioned, initialization is handled through the calling of a constructor. Java is the only language that allows a default constructor without the need to type it. C++ and Python do allow the use of default constructor if typed. A default constructor in most languages is like this piece of code `Example().` Java, C++ and Python allow the use of a constructor with parameters. An example of this would be `Example(Qts Q, int Pts, String examName).` Fortran 2003 in addition to allowing a constructor with parameters, also allows constructors to have procedures in them. A procedure is a function such as the area of a circle. Common Lisp constructors are handled differently in that they take the form for a list where the parameters are declared first and then each parameter is loaded with its values for use later, like this `(capacity 3) (carType '(Truck SUV Van SportsCar))).` Finally, simula constructors are typed by setting the variables and then declaring their type, like this `Class Triangle(Base, Height); int Base, Height;`. Once all of the programming languages have allowed for the creation of objects, how are they removed? That would be through the process of finalization. In the case of Java, Python and Common Lisp finalization is accomplished through the use of garbage collection, which is where language automatically removes objects that are no longer in use. In the case of C++ and Fortran 2003 the finalization of objects occurs through the use of destructors, in C++ a destructor would be calling the delete on a variable or object. Finally, the language simula is so old it does not use finalization in a modern sense, but instead uses the keyword end to let the computer know that an instance of an object is done being used. Initialization and finalization, while invaluable in the programming domain, are not without their challenges.

Challenges

As mentioned above, there are challenges that come along with implementing initialization in a programming language. The first is choosing a constructor. In object-oriented programming languages, languages may allow zero, one, or more constructors for a given class. Other languages have dealt with this problem by allowing multiple constructors for a class that takes different parameters. In C++, Java, and C#, the constructors act like overloaded subroutines. Another way is, changing the name of the constructor method in some pattern according to the language specifics used in languages like Smalltalk and Eiffel. Examples include incrementing a number in the namespace of the method.

Another challenge facing programmers is values and references. An object can be either referenced by a variable, or the value of the variable is the object. Languages such as Smalltalk, Ruby, Simula, Python, and Java use variables to reference an object. This means every instance of an object must be explicitly created. On the other hand, some languages allow a variable to have a value that is an object. These languages include C++ and Ada. When derived classes are used, the compiler must know the execution order of constructors to call to create that derived object. Languages like C++ require every object to be initialized before the variable or the object can be used. This way constructor calls can be made on derived classes.

The last challenge that warrants discussion is garbage collection. In languages that do not have an automatic garbage collection system like C++, the use of destructors is used to free up the space. If destructors are not used, the program can crash from running out of space and not efficiently using memory. In languages like java and python, garbage collection is automatically implemented and reclaims memory at runtime. Garbage collection deals with any unreferenced code allowing the memory to be dynamically updated with new information. Despite there being challenges to using initialization and finalization there are some positive attributes of their use.

Pros of Initialization & Finalization

Initialization and finalization are essential to object-oriented programming and the main pro of initialization is to assign objects and variables space in memory. Assigning objects and variables space in memory ensures fast performance of a program by only using up the required memory. This is superior to languages such as scheme or prolog that use additional memory at every step in their program's process. A second pro of initialization is requiring the programmer to decide the values assigned to the objects and variables, which in turn means the results of the program will be the outcome desired. The pro of finalization is to free up memory that is no longer being used by an object. This is a good process that prevents resource intensive objects from using up all of the memory available in a computer. Without finalization it would be impossible for other parts of the program to be completed. Overall, the pro of initialization and finalization is in the use of resource management of memory in a computer. While most computers nowadays will never have a program that uses up all its memory in the past this was a very big issue. Initialization and finalization does have its pros but, there are also cons.

Cons of Initialization & Finalization

In some coding languages, the use of garbage collection is automatic and doesn't require a low level of memory management. However, an object-oriented programming language like C++ requires the programmer to have knowledge of how to deal with garbage. In most cases, users will implement smart pointers to deal with unreferenced objects. This way, the use of a destructor is not needed as the smart pointers realize when they have no reference then remove themselves in memory. If smart pointers are not used, the destructors will need to be created to remove the unwanted object after being done with its lifetime. If the programmer is not conscious and considerate of the destructors, the program will run out of memory

Conclusion

Initialization and finalization play a significant role in most all programming languages. Through different implementations, programming languages have used these constructs to create objects and variables, as well as delete them. Object oriented programming languages are the most popular types of languages in this day and age. Their use of initialization and finalization has historical roots, some challenges but mostly positive attributes. It is safe to say without initialization and finalization object-oriented languages would not be able to accomplish their tasks.

References

Corob-Msft. (2021, August 3). *Destructors (C++)*. Microsoft Docs. Retrieved November 19, 2021, from https://docs.microsoft.com/en-us/cpp/cpp/destructors-cpp?view=msvc-170.

cs.cmu.edu. (n.d.). *19.6. By-Position Constructor Functions*. Common Lisp the Language, 2nd Edition. Retrieved November 19, 2021, from https://www.cs.cmu.edu/Groups/AI/html/cltl/clm/node1.html.

Danny, V. (2019, October 24). *Tutorial oop(iii): Constructors and Destructors*. The Delocalized Physicist. Retrieved November 19, 2021, from https://dannyvanpoucke.be/oop-fortran-tut4-en/.

DeBrie, A. A. (2021, May 3). *Python garbage collection: What it is and how it works*. Stackify. Retrieved November 19, 2021, from https://stackify.com/python-garbage-collection/.

Furuknap, B. (2015, December 4). *How to: Object-oriented programming - atlantic.net*. Atlantic.Net Blog. Retrieved November 19, 2021, from https://www.atlantic.net/vps-hosting/how-to-object-oriented-programming-classes-objects/.

Singh, C., says, P. J., Jakhar, P., & says, C. S. (2018, March 10). *Python constructors - default and parameterized*. beginnersbook.com. Retrieved November 19, 2021, from https://beginnersbook.com/2018/03/python-constructors-default-and-parameterized/.

Sklenar, J. (1997, December 5). *INTRODUCTION TO OOP IN SIMULA*. Introduction to Simula. Retrieved November 19, 2021, from http://staff.um.edu.mt/jskl1/talk.html.

Tutorialspoint. (n.d.). *https://www.tutorialspoint.com/java/lang/object_finalize.htm*. Java.lang.Object.finalize() method. Retrieved November 19, 2021, from https://www.tutorialspoint.com/java/lang/object_finalize.htm.

The User's Guide - 9 SEP 1996. (n.d.). *5.1 About storage in Lisp*. 5 Storage Management in Common Lisp. Retrieved November 19, 2021, from http://www.lispworks.com/documentation/lcl50/ug/ug-67.html.