Object-Oriented vs. Imperative Programming
Topic Paper #12

Alex Laird
CS-3210-01
2/27/09
Survey of Programming Languages
Spring 2009
Computer Science, (319) 360-8771
alexdlaird@cedarville.edu

Grading Rubric		Max	Earn
	On Time/Format	1	
	Correct	5	
	Clear	2	
	Concise	2	
	Total	10 pts	

Peer Reviewer

ABSTRACT
This paper describes structural concepts of object-oriented programming in comparison to imperative programming.

Keywords
Object-Oriented, Imperative, Procedural, Programming

1. INTRODUCTION
The first programming languages, languages such as FORTRAN and ALGOL, were imperative, procedural programming languages. As programs became more procedural and more modular, object-oriented design became a more intriguing and versatile conversation, and the progression from these paradigms is what has helped define modern programming.

2. IMPERATIVE PROGRAMMING
FORTRAN, developed in 1954, was the first major imperative programming language. Since early imperative instructions tended to be very simple, communication with the archaic (and not versatile) hardware was easy.

The basics of imperative programming are the simplest foundations of all programming paradigms: expressions and commands [2]. Imperative programming languages are known for the simple, straightforward approach to problems [3].

3. THE PROCEDURAL EXTENSION
While early imperative programming languages did allow procedures (sometimes called subroutines or functions), these procedures were very simple computations and quickly returned to the main routine. However, at some point imperative languages became much more reliant on these procedural extensions and actually allowed computations and states to be localized to a procedure, thus gaining the name "Structured Programming," which improved maintainability immensely [2].

It was these procedural extensions that led to the birth of object-oriented programming.

4. OBJECT-ORIENTATION
Object-oriented programming took procedural programming to the next level, making entire classes to store an object and procedures within that object to manipulate it. The first major object-oriented breakthrough was in the PDP-1 computer at MIT in 1960, but the first major programming language to be called "Object-Oriented" was Simula in 1969.
In reality, object-oriented concepts are simply large-scale extensions of imperative and procedural programming designs, though they are highly useful and versatile extensions.

5. CONCLUSIONS
Object-oriented programming is the brain-child of imperative programming and would not exist without its parent. On that note, we would not have the large-scale, highly modular, highly powerful programming abilities that we have today without the growth of imperative procedural programming into what is now object-oriented programming.

6. REFERENCES
[1] Sebesta, Robert W., 2008. Concepts of Programming Languages. Addison-Wesley, Boston. ISBN: 978-0-321-49362-0

[2] Reddy, U. S. 1996. Imperative functional programming. *ACM Comput. Surv.* 28, 2 (Jun. 1996), 312-314. DOI= http://doi.acm.org/10.1145/234528.234736

[3] Gifford, D. K. and Lucassen, J. M. 1986. Integrating functional and imperative programming. In *Proceedings of the 1986 ACM Conference on LISP and Functional Programming* (Cambridge, Massachusetts, United States). LFP '86. ACM, New York, NY, 28-38. DOI= http://doi.acm.org/10.1145/319838.319848

Alex Laird, Cedarville University, Cedarville, Ohio, 45314