Role of the C language in Current Computing Curricula Part 1 – Survey Analysis

Alessio Gaspar,
Alessio@lakeland.usf.edu
University of South Florida Lakeland
3433 Winter Lake Rd, Lakeland 33803, Fl, USA

Abdel Ejnioui
aejnioui@lakeland.usf.edu
University of South Florida Lakeland
3433 Winter Lake Rd, Lakeland 33803, Fl, USA

Naomi Boyer
nboyer@lakeland.usf.edu
University of South Florida Lakeland
3433 Winter Lake Rd, Lakeland 33803, Fl, USA

Role of the C language in Current Computing Curricula Part 1 – Survey Analysis

Abstract

In December 2006, an anonymous online survey was publicized on the various ACM mailing lists (SIGCSE, SIGITE). Its purpose was to determine the role of the C language in the various modern computing curricula (CS, IT…). This paper summarizes the results and stresses out the quantitative usage of this language in introductory and intermediate programming courses as well as in upper-level undergraduate courses (e.g. operating systems). We also present the qualitative reasons provided by our respondents for, or against, the adoption of the C language in these various courses. We then discuss these results and propose an analysis of when the C language might be most useful in the curriculum, how it should be introduced and what specific topics should be covered in such a re-designed "intermediate programming in C" course.

1. Introduction

From a technical and historical perspective, the impact of the C language on computing disciplines is not one to be overlooked easily. However, from a pedagogical perspective, C has never been perceived as a language of choice for introducing students to programming. While other popular educational and professional languages, such as Java, also have significant pedagogical issues **[1]**, the computing education communities have invested significant efforts in developing suitable environments for students to learn with **[2, 3, 4, 5, 9, 10]**. Such efforts are clearly few on the C front **[8, 11]** for which the computing education community seems pretty much to have given up. This observation prompts two questions; can we facilitate the learning of C through use of appropriate IDEs or pedagogies and can we leverage C's characteristics in another niche than introductory programming?

The following sections will focus on presenting observations resulting from analyzing the data of our online survey. Section #2 will focus on introducing the survey and analyzing the respondents' profile in order to flesh out some context in which to interpret the observations. Section #3 will present observations related to the choice of programming languages expressed by the respondents as far as introductory or intermediate programming courses as well as upper-level undergraduate courses of the curriculum. Sections #4 and #5 will respectively review the arguments against or in favor of the adoption of the C language in the above-mentioned levels. Section #6 will conclude by reviewing how C is used in upper-level courses. This paper will be continued in its second part which will analyze further the observations detailed herein.

2. Survey & participants description

The objective of this study is to assess the role played by the C programming language in the modern computing curricula (CS, IT, CE, CIS…) in higher education. Specifically, our focus is on evaluating its usage from two complementary perspectives: a quantitative review of the number of instructors using it as well as qualitative perspective identifying its weaknesses and strengths. The use of C is explored at three different levels: introductory programming courses (e.g. CS-1), intermediate programming courses (e.g. CS-2 and above) and other upper-level undergraduate courses of the curriculum (e.g.

operating systems and networking, etc.). The findings go beyond the expected rejection of C as a language for beginners in order to identify its useful and active niches. We used an anonymous online survey to collect quantitative and qualitative data. Questions were designed to explore the pros and cons of C in each specific course level as well as to evaluate qualitatively the key language features. This analysis was used to identify the technical reasons for accepting or rejecting C and discuss whether they could be addressed by pedagogical or technological solutions. The survey was posted online using surveymonkey.com and included 4 parts: general information, usage of C in introductory programming courses, in intermediate ones, in upper-level undergraduate ones. Calls for participation were posted on both ACM's special interest groups in computer science education (SIGCSE) and information technology education (SIGITE) mailing lists. After two weeks, we had 97 respondents representing the following disciplines (multiple choices allowed); 90.6% Computer Science, 9.4% Information Technology, 4.2% Computer Engineering and 5.2% Computer Information Systems. In addition, 2.1% selected "Software Engineering" and "Computational Science" as "other". Concerning institutional profiles, 47.4% indicated being in an undergraduate + graduate university, 43.2% in an undergraduate only institutions and 4.2% in a community college. These numbers were instrumental in understanding the context of the survey's findings which were also influenced by the demographics of both SIGCSE and SIGITE mailing lists. Most responses were received shortly after the SIGCSE announcement and before the SIGITE one thus making results mostly CS-centric.

3. Programming languages choices

The second and third parts of the survey focused respectively on the usage of the C language in introductory and intermediate programming courses. Our survey also inquired about the languages used in upper-level undergraduate courses (e.g. operating systems, networking, software engineering, etc). For each course level, respondents were allowed multiple choices among 12 languages and an "other" category (c.f. Table 1).

Language	**Introductory courses**		**Intermediate courses**		**Other courses**	
	Respondents		**Respondents**		**Respondents**	
	%	**#**	**%**	**#**	**%**	**#**
Java	72.0	67	73.9	68	82.4	75
C	14.0	13	10.9	10	**67.0**	61
C++	9.7	9	32.6	30	69.2	63
Python	4.3	4	5.4	5	20.9	19
Visual Basic	4.3	4	2.2	2	9.9	9
C#	3.2	3	5.4	5	15.4	14
Please refer to **[25]** for full list of languages						
Others	10.8	10	4.30	4	20.9	19
Respondents Total		**93**		**92**		**91**

Table 1: Languages used in introductory and intermediate programming courses.

The response rate for this question was high; out of 97 respondents, 93 responded about introductory courses, 92 about intermediate ones, and 91 for other upper-level ones. The results revealed that Java dominates at all course levels (from 72% to 82.4%). This is consistent with its popularity among instructors and professionals alike and is also linked to the rise of the object-first approach **[1]**. In contrast, the C language only came as a distant second choice in introductory courses (14%), a third choice for intermediate ones (10.9%) and a third choice in the rest of the curriculum (67% for C, 69.2% for C++, 82.4 for Java). C is clearly perceived as necessary for some upper-level courses, but not as a language for beginners. This rejection can be also seen as influenced by the nation-wide decline of enrollment in computing curricula which caused the computing education communities to develop environments dedicated to CS-1 **[2, 3, 4, 5]** along with innovative pedagogies relying on higher level languages. These CS-1 approaches are as far from C as can be; they are meant to be visually attractive, pleasing to the beginning student in order to attract a larger population of students to the computing field.

Among the languages mentioned in the "other" category, most are functional: Lisp, Haskell, ML and Scheme (5 mentions at introductory level, 6 in other courses).

4. Rejection of the C language

Negative Qualities of the C Language	**Introductory Courses**		**Intermediate Courses**	
	Respondents		**Respondents**	
	%	**#**	**%**	**#**
Lack of object-oriented features	72.6	61	79.7	63
Difficulties related to explicit memory allocation	46.4	39	34.2	27
Difficulties related to pointers arithmetic	42.9	36	27.8	22
Other (please specify)	40.5	34	38.0	30
Unsuitability of the general syntax (if, for, curly braces...) although compatible with other language	14.3	12	7.6	6
Variables declaration syntax (right-left reading)	6.0	5	2.5	2
Respondents Total		**84**		**79**

Table 2: Reasons for not using the C language.

The survey solicited qualitative justifications of why C was not used in introductory and intermediate programming courses by using both multiple choices and an open question. The results (cf. table 2) clearly identify the lack of object oriented features as the most important reason to reject C in both introductory (72.6%) and intermediate (79.7%) programming courses. This concern even took precedence over technical reasons; pointer arithmetic detracted instructors from using C in introductory (42.9%) courses rather than intermediate ones (27.8%) while explicit memory allocation was more heavily criticized at both levels (46.4% introductory vs. 34.2% intermediate). The results also revealed that the C syntax is no longer perceived as a pedagogical hindrance with the exception of a small number of respondents (14.3% introductory vs. 7.6% intermediate). This contrasts with previous work **[7]** but can be explained by the adoption of other C-like languages (C++, C#, Java). For completeness, an "other" category allowed participants to provide alternative reasons for not using C in introductory and intermediate programming courses **[6]**. The following themes emerged from these responses.

Impact of C Features on Pedagogy; The responses overwhelmingly criticized C for its lack of object oriented features. However, other critics were formulated concerning the lack of real type safety, the lack of a proper string data type, poor runtime error detection and reporting, and awkward I/O. These shortcomings translate into pedagogical difficulties which all revolve around the idea that C is too much of a “low level” language which exposes students to intricate topics which distract them from problem-solving and zero-defect programming skills. Along this theme, two specific comments stand out: “far too complex for our students” and “C is a cruel thing to inflict on beginners”.

The “Java Bandwagon”; one of the respondents coined up this term to describe the position of others on the Java adoption issue. This theme emerged from responses which stressed the need to use Java because it has become an educational or industrial standard without need for further technical or pedagogical justification. Variants included the need to be “compatible” with other universities or AP exams, to easily accommodate transfer students. Java might have reached a critical mass turning it into the *de facto* language for programming education, however, it now shadows alternatives such as Scheme or Python which have been argued to be better suited for the task, given the complexity of Java **[1]**. As the survey revealed, these alternative are in use by a minority of participants only.

5. Adoption of the C language

In order to help us identify aspects of C which are inappropriate for beginners but worth teaching to intermediate students, we used very similar wording in the questions prompting the respondents for reasons for which they are using the C language.

Positive Qualities of the C language	Introductory courses		Intermediate Courses	
	Respondents		Respondents	
	%	#	%	#
C is used in advanced courses to which students need to be prepared for	50.0	10	41.1	8
C exposes low-level concepts (stack, variable allocations classes...) useful in other courses	45.0	9	64.7	11
Lack of Object Oriented features allows for a better focus on fundamentals	40.0	8	5.9	1
General Syntax is compatible with other languages	35.0	7	23.5	4
Pointers / Explicit memory management teach students about useful low level concepts	30.0	6	58.8	10
C helps students acquire a strong programming discipline	30.0	6	5.9	1
C serves as a selection tool to identify strong candidates to enter your curriculum	0.0	0	11.8	2
Other (please specify)	40.0	8	35.3	6
Respondents Total		**20**		**17**

Table 3: Reasons for using the C language.

Because C is not used by many participants in their programming courses, the total number of respondents for this question has been low for both introductory (20 respondents) and intermediate (17 respondents) courses (c.f. Table 3). At introductory level, the faculty who responded to these questions clearly indicated that the lack of object oriented features and the necessity to understand low level programming concepts were actually desirable for intermediate-level programming courses. This makes sense in so far that about half of the respondents (50%, 41.1%) indicated that they were using the C language because it is pre-requisite to upper-level courses of their curriculum.

C is being used and adopted in intermediate courses to prepare students for more advanced coursework (41.1%). However, its potential for exposing students to both pointers (58.8%) and low-level programming concepts (64.7%) takes precedence. Interestingly, these results indicate how the very aspects of C which are perceived as a pedagogical hindrance in introductory courses can be useful to provide a more in-depth understanding of programming at later stages of student education. Although small in numbers at the moment, some educators recognize the acquisition of these peculiar aspects of the C language as an integral part of their curricular teaching approach. As of responses in the "other" category, the following themes emerged:

Curricular Needs; In support of the figures from Table 3, a comment by a respondent indicated that C was used in CS-3 to prepare students for courses such as operating systems, computer architecture, and networking. C has also been noted to be used in computer organization courses or special sections in which it plays a role very similar to assembly languages a decade ago. These comments further support the finding of a restricted niche of educators who deem the C language suitable for some specific topics.

Industry Needs; The second emergent underlines the fact that some institutions feel the need to cover C in their curriculum due to its presence in some industries such as embedded systems, security, and computer engineering and device-level development.

6. C in other courses

Previous parts of this survey focused on introductory and intermediate programming courses. The last part was designed to evaluate the use of C in other courses of the curriculum. While the previous findings indicated that C is appropriate in system-level courses, the following results show, which courses are using C as a pre-requisite.

	Respondents	
Other courses using C	**%**	**#**
Operating systems	51.6	47
Networking	24.2	22
Computer organization	16.5	15
Data structures	6.6	6
Software engineering	1.1	1
None	28.6	26
Others (please specify)	26.4	24
Respondents		**91**

Table 4: C in the rest of the curriculum.

Table 4 clearly indicates that 28.6% of our respondents do not use C in any other course. The remaining respondents use C in operating systems (51.6%), networking (24.2%) and computer organization (16.5%). The "other" category (26.4%) revealed that C was also used in compilers, embedded systems, and mainly in Unix system programming courses. To probe further, participants were invited to provide reasons for which they adopted C in these courses as a list of up to nine items. Analysis revealed that these responses can be divided into the five categories appearing in table 5. In order to weigh their relative consequence, each response comment was assigned to one of these categories and its importance weighted with a researcher designed point system. Each time a response from the first line was attributed to a category, the latter was awarded seven points. When a response was cited as second within a category, six points were awarded to the response. This was repeated until responses given as fourth choice or below were reached. Since the numbers for the latter were low, they were lumped together. Table 5 shows the resulting categories and their global weight.

Feedback Category	**Global Weight**	**Detail Weight**			
		7	**6**	**5**	**4,3,2,1**

Operating systems/System programming	185	21	2	4	1.0
Computer architecture	149	13	3	5	1.2
Technical aspects of C	112	9	4	4	2.0
Exposure to C	77	4	6	0	1.2
Other courses	53	5	3	0	1.2

Table 5: C in the rest of the curriculum.

The figures in Table 5 indicates that most comments support the findings mentioned above to the extent that operating systems and system programming courses are the primary courses using C, with architecture courses coming as second. The responses also illustrate that a significant number of adoptions are motivated by either specific qualities of the language or a justified need for exposure to programming in C for some curricula. Specifically, this need for exposure is motivated by various considerations such as providing students with experience on a diverse set of languages throughout the curriculum, historical ties to the Unix operating systems, prevalence in some specific industrial fields, and "reinforcement of students learning". Several comments also contributed to the results of Table 5 by indicating that C was used to be compatible with existing tools.

7. References

[1] Roberts, E.; et al. The ACM Java Task Force: Final Report. 37th SIGCSE technical symposium on Computer Science Education; Houston, Texas. ACM Press; 2006. pp. 131-2.

[2] Kolling, M., Quig, M., A. Patterson, J. Rosenberg, "The BlueJ system and its pedagogy", Journal of CS Education, special issue on learning and teaching object technology, vol 13, no 4, 12/2003

[3] Dann, W., Pausch, R., Alice: a 3-D tool for introductory programming concepts, Stephen Cooper, Journal of Computing Sciences in Colleges , Proceedings of the fifth annual CCSC northeastern conference on The journal of computing in small colleges CCSC '00, Volume 15 Issue 5, 2000

[4] Külling, M., Henriksen, P., Game programming in introductory courses with direct state manipulation, June 2005, ACM SIGCSE Bulletin , Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education ITiCSE '05, Vol. 37:3

[5] Guzdial , M., Use of collaborative multimedia in computer science classes, June 2001, ACM SIGCSE Bulletin , Proceedings of the 6th annual conference on Innovation and technology in computer science education ITiCSE '01, Volume 33 Issue 3

[6] Gaspar, A., Ejnioui, A., Boyer, N., Role of the C language in modern computing curricula, Survey monkey results http://surveymonkey.com/DisplaySummary.asp?SID=3039744&Rnd=0.9342091

[7] Canning, J., Moloney, W., Rafyemehr, A., Rey, D., Reading types in C using the right left walk method, June 2004, ACM SIGCSE Bulletin , Working group reports from ITiCSE on Innovation and technology in computer science education ITiCSE-WGR '04, Volume 36 Issue 4

[8] Stephen, N., Freund, Roberts, E,S., Thetis: an ANSI C programming environment designed for introductory use, March 1996, ACM SIGCSE Bulletin , Proceedings of the twenty-seventh SIGCSE technical symposium on Computer science education SIGCSE '96, Volume 28 Issue 1

[9] Cross, J.H., jGRASP: an integrated development environment with visualizations for teaching Java in CS1, CS2, and beyond, April 2006, Journal of CS in Colleges, Volume 21 Issue 4

[10] Allen, E., Cartwright, R., Stoler, B., DrJava: a lightweight pedagogic environment for Java, February 2002, ACM SIGCSE Bulletin , Proceedings of the 33rd SIGCSE technical symposium on Computer science education SIGCSE '02, Volume 34 Issue 1

[11] Demetrescu, C.; Finochi, I., Leonardo: A C programming environment for reversible execution and software visualization. [Web Page] 1999; http://www.dis.uniroma1.it/~demetres/Leonardo/. [Accessed Apr 2006].