http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

Information Technology Journal 7 (7): 1055-1060, 2008
ISSN 1812-5638
© 2008 Asian Network for Scientific Information

Developing an Intelligent Tutoring System For Students Learning To Program in C++

Samy S. Abu Naser
Faculty of Engineering and Information Technology, Al-Azhar University, Gaza, Palestine

Abstract: The aim of this study is developing an intelligent tutoring system for helping students enrolled in computer sciences 1 (an introductory C++ programming course) at the Faculty of Engineering and Information technology in Al-Azhar University. The C++ Intelligent Tutoring System is called CPP-Tutor. In this paper we present an overview of the CPP-Tutor architectural design and user interface. This pilot project is for constructing a model domain of a subset of the C++ programming language. The completed project will be sufficient to prove the concept and that a fully developed C++ Intelligent Tutoring System will provide an interactive learning environment for students. According to the success of other similar Intelligent Tutoring Systems, it is also hypothesized that students will be able to learn to program in C++ and gain knowledge more quickly and effectively than students using traditional methods of teaching.

Key words: Artificial intelligence, intelligent tutoring system, C++, programming, e-learning

INTRODUCTION

An Intelligent Tutoring System (ITS) provides individualized computer-based instruction to students (Abu Naser, 2006, 2001; Brusilovsky *et al*., 1996). These systems emerged from application of artificial intelligence techniques to the Computer Aided Instruction (CAI) systems (Abu Naser and Sulisel, 2000). The difference is that an ITS usually compares the student's work with expert solutions or strategies, models the student's probably knowledge of a domain and provides coaching or advice, taking into account what the student's knowledge state and preferred learning style. Depending on Artificial Intelligence and cognitive science, ITS became very popular and effective domain in Education for many reasons: Better student performance, student learns in less time and student is in the driver seat (Anderson *et al*., 1995; Woolf *et al*., 2001; Graesser and Parson, 2001). For many years, there is a continuous development and evaluation of ITS (Shute and Glaser, 1990) for tutoring and monitoring programming in the field of artificial intelligence in education. Programming requires a group of problem-solving and diagnostic strategies. The behavior in which a student writes code provides great insight into the way of his thinking. As a result, programming provides attractive area to study learning and cognitive processes (Koedinger, 2001). Among the objectives of this undergoing research is to gather the developments in the ITS, Cognitive Science and AI to make a useful intelligent tutor to: help students learning to program in C++ and provide personalized instruction for students learning to program in C++. Furthermore, it is hoped that this research will have a positive impact on supporting instructors teaching C++ programming in their classroom.

Tutors related to CPP-Tutor have been developed to teach programming languages like: PROUST (Johnson and Proust, 1985) and LispTutor (Anderson and Skwarecki, 1986), MENO-II (Soloway *et al*., 1983) and ELM-PE (Weber and Möllenberg, 1995). They involved students in entering code to solve a given problem. PROUST attempted to estimate the students' plan intentions, while LispTutor guided their left-to-right, top-down attempts in a highly directed fashion by interpreting their code as a correct or a buggy solution. MENO-II and ELM-PE analyze the user's solutions to exercises and provide feedback to identified misconceptions or missing skills based on the analysis. Other approaches have also emerged in intelligent tutoring paradigm. Kumar's model-based tutor asks students to predict C++ programs' output and identify semantic and run-time errors (Kumar, 2002). It provides explanations of program execution line by line to help students understand code behavior. Adaptive navigation based on student modeling is used in a web-based system called ELM-ART II (Weber and Specht, 1997) to provide individualized annotated hyperlinks and curriculum sequencing. Expert critiquing systems provide useful feedback to users' work in many domains. Such feedback includes alerts to problematic situations, relevant information to the task at hand, directions for improvement and prompts for reflection (Silverman, 1992). One of these systems is Java Critiquer that uses an incremental authoring approach to amortize the high development cost (Qiu and Riesbeck, 2004).

Problem: Write a program in C++ which find and print the cubic value for each integer between -5 and +5.

Program specifications: This program can be written using for or while loop.

The solution which is authored by Instructor:
```
#include<iostream.h>
void main() {
    int c, j= -5;
   while (j <= 5) {
      c = j * j * j;
     j++;
      cout << c << endl;
   }
 }
```
The student first possible error : (missing #include<iostream.h> in line 1)
```
void main (){
    int c, j= -5;
   while (j <= 5) {
      c = j * j * j;
     j++;
    cout << c << endl;
   }
 }
```
The student second possible error: (void is missing in line 2)
```
#include<iostream.h>
main () {
   int c, j= -5;
   while (j <= 5) {
   c = j * j * j;
   j++;
   cout << c << endl;
    }
 }
```
The student third possible error: (unknown Int in line 3)
```
#include<iostream.h>
void main() {
  int c, j= -5;
  while (j <= 5) {
  c = j * j * j;
  j++;
  cout << c << endl;
   }
 }
```
The student fourth possible error: (undeclared variable c in line 5)
```
#include<iostream.h>
void main() {
  int j= -5;
  while (j <= 5) {
  c = j * j * j;
  j++;
 cout << c << endl;
  }
 }
```
The student fifth possible error: (the body of the loop is never executed)
```
#include<iostream.h>
void main() {
  int c, j= -5;
  while (j >= 5) {
  c = j * j * j;
  j++;
  cout << c << endl;
  }
 }
```

Fig. 1: Continued

The student sixth possible error: (last value in the range is not calculated, i.e., '+5')
```
#include<iostream.h>
void main() {
  int c, j= -5;
  while (j < 5) {
  c = j * j * j;
  j++;
  cout << c << endl;
   }
 }
```
The student seventh possible error: (the output of the last value of c printed only)
```
#include<iostream.h>
void main() {
  int c, j= -5;
  while (j <= 5) {
  c = j * j * j;
  j++;
 }
  cout << c << endl ;
 }
```

Fig. 1: A problem example with a solution and a few errors

Other systems like CTutor (Song *et al*., 1997), Prolog Tutor (Hong, 2004) and Java Intelligent Tutoring System" (Sykes and Franek, 2003) adopts advanced code analysis to see what is the intention of student and give feedback based on this analysis.

Architecture of CPP-tutor: We present the model and architecture for the C++ intelligent tutoring system, the knowledge base design, expert module design, feedback module design and the user interface design.

CPP-tutor knowledge base design: Here we describe the Knowledge Base architectural module for CPP-Tutor. In the current pilot project we use a subset of C++ programming language grammar to be tutored. We have concentrated on the following topics of the standard C++ language (Alexandrescu, 2001): variables declaration, operators, assignments and looping (for loop and while loop) structures.

For every problem stored in the database, we have stored one possible solution, a few possible errors for specific categories and a few possible hints for each error. Figure 1 shows a problem example with a solution and some errors.

There are many possibilities for student answers and the system cannot simply list all incorrect answers of the student. For instance, the student could write:

c = pow(j, 3); or c = pow(j, 2)* j

Both answers are completely correct and the system needs to recognize these types of answers and not treat them as incorrect answers. Testing the correctness of a program is a hard task. CPP-Tutor is designed to be pedagogically sound (Sykes and Franek, 2003). So, although the above formulas result in correct answers, this is not the only goal of the tutoring system. Rather, CPP-Tutor focuses on the methodology by which a student attempts to solve a problem. Just as presented in this example, CPP-Tutor is focusing the student on the problem on hand by specifying the location where code may only be written. Conventions, style and professional programming techniques are modeled in CPP-Tutor. In this fashion effective tutoring may take place.

CPP-tutor expert module: In order for CPP-Tutor to provide intelligent feedback to the student the Expert Module relies on a group of information: the problem statement, the problem specification, student's code, the established student model, the C++ compilation and the result from the C++ runtime engine. Based on this context, some of this information will not be on hand. However, the goal of this module is to carefully analyze all available information so that appropriate feedback may be generated for the student. This is accomplished by the core component of the Expert Intent Recognition Module.

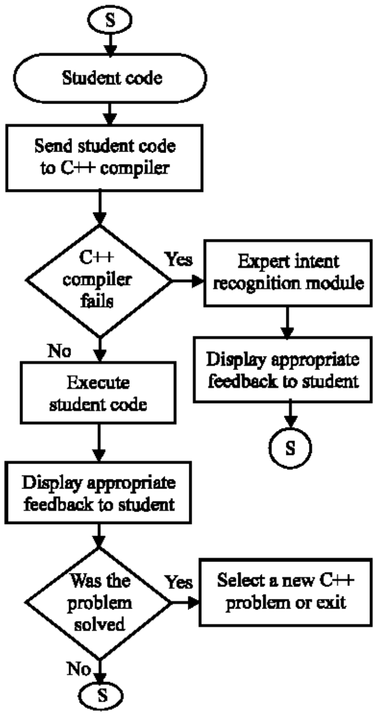
Expert intent recognition module: The purpose of the Expert Intent Recognition (EIR) Module is to make sure that the most plausible submission of code that the student intended. As shown in Fig. 2, the EIR is invoked when the standard C++ compiler fails. Assume that the submitted student solution is called Sol1 and the existing solution for the given problem is Sol2.

The EIR module performs pattern matching between the two solutions (Sol1 and Sol2) to generate a transformation function string and to calculate the edit distance between the student's solution and the actual solution using a Dynamic Programming Algorithm (DPA).

The EIR module constructs a transformation mapping function string (i.e., T: Sol1 ➔ Sol2). T involves all insertions, deletions, transpositions and character changes that are required to transform Sol1 into Sol2. The cost for an insertion, deletion, transposition, or character change is 1. The EIR module then constructs feasibly-sound variations (i.e., modified Sol1) of the student's solution and proceeds to compile and run them. For example, let the student's submitted solution (Sol1) be:

```
"#include<iostream.h>
void main( {
Int c,j = -5
while(j>=5){
c: = j*j*j;j++;cout <<c<<endl;}}"
```

and the actual solution (Sol2) be:



S
Student code
Send student code to C++ compiler
C++ compiler fails
Yes
Expert intent recognition module
Display appropriate feedback to student
S
No
Execute student code
Display appropriate feedback to student
Was the problem solved
Yes
Select a new C++ problem or exit
No
S

Fig. 2: Flow chart for CPP-tutor

```
"#include<iostream.h>
void main(){
int c,j = -5;
while(j< = 5){
c = j*j*j;j++;cout <<c<<endl;}}"
```

The distance between Sol1 and Sol2 is 5 and
T = "♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
♦♦♦♦♦♦♦♦♦♦β♦
δ♦♦♦♦♦♦♦♦♦β
♦♦♦♦♦♦♦η♦♦♦♦
♦ζ ♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦",

where the symbols ♦, β, δ, η and ζ means: match, insertion, case mismatch, replacement and deletion, respectively. Furthermore the suggested modified student solution as follows:

```
Sol1 ="#include<iostream.h>
void main(β {
int c,j = -5β
while(j< = 5){
c = j*j*j;j++;cout <<c<<endl;}}"
```

The new Sol1 portray the pattern matching procedure. The EIR module passes this information on to the Expert module to determine the appropriate response for the student.

```
1. #include<iostream.h>
2. void main( {
3. Int c,j=-5
4. while(j>=5){
5. c:=j*j*j; j++; cout <<c<<endl; }}
```

CPP-Tutor: "It seems something is missing in line 2. It might be ')' between '(' and '}' Do you agree?"
Student: Yes
CPP-Tutor makes the correction and proceeds with rest of the code.

```
1. #include<iostream.h>
2. void main() {
3. Int c,j=-5
4. while(j>=5){
5. c:=j*j*j; j++; cout <<c<<endl; }}
```

CPP-Tutor: "I suppose that you need the keyword 'int' instead of 'Int'. Is that what you meant?"
Student: Yes
CPP-Tutor makes the correction and proceeds with rest of the code.

```
1. #include<iostream.h>
2. void main() {
3. int c,j=-5
4. while(j>=5){
5. c:=j*j*j; j++; cout <<c<<endl; }}
```

CPP-Tutor: "It seems something is missing in line 3, It might be ';' Do you agree?"
Student: Yes
CPP-Tutor makes the correction and proceeds with rest of the code.

```
1. #include<iostream.h>
2. void main() {
3. int c,j=-5;
4. while( j>=5){
5. c:=j*j*j; j++; cout <<c<<endl; }}
```

CPP-Tutor: "The symbol '>' found in line 4, It should be '<' instead'. Is that what you meant?"
Student: Yes
CPP-Tutor makes the correction and proceeds with rest of the code.

```
1. #include<iostream.h>
2. void main() {
3. int c,j=-5;
4. while( j<=5){
5. c:=j*j*j; j++; cout <<c<<endl; }}
```

CPP-Tutor: " The symbol ':=' found in line 5, It should be '=' instead'. You are learning C++ not Pascal Language. Is that what you meant?"
Student: Yes

Fig. 3: Dialogue between CPP-Tutor and Student

Before the EIR module makes any changes as indicated in the T function string, the student is asked specific questions for intent clarification purposes. In this fashion, only modifications that the student knows can be made to the code. So, using the example in Fig. 1, the following dialogue between CPP-Tutor and the student would come up as in Fig. 3:

The tutoring process is dynamic. The student may at any time disagree with CPP-Tutor suggestions and modify the source code. After the modification of the user, tutoring process begins again.

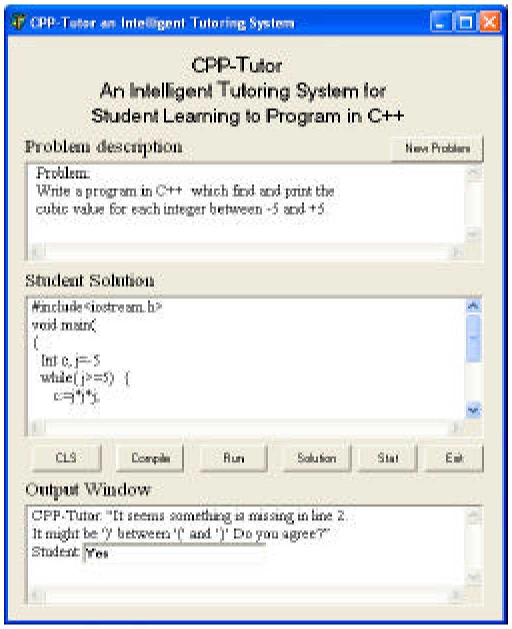
CPP-tutor feedback module: The effectiveness of any Intelligent Tutoring System depends heavily upon its feedback timing and style. Timing refers to when the student is given a response to the solution. When the feedback is presented to the student should be governed by what the student knows. For example, in the beginning of programming tutoring, the tutor may wish to correct all of the syntax errors the student is making as these mistakes are made. As the student progresses and learns more, the feedback about the syntax of the language would be annoying. Instead, the system should present immediate feedback for the higher level concepts that the student is attempting to learn and delay the feedback for the lower level concepts. Tutors are better than teachers in this respect in that they can provide a student with timely feedback better than most teachers (Mark and Greer, 1995).

CPP-tutor user interface: The interface for computer-based programming tutoring system is a very important factor that we gave it a careful consideration during the design of CPP-Tutor (Koedinger, 2001). The user interface is based on a presentation format implemented in many popular Integrated Development Environments used by professional programmer (Abu Naser, 2008; Conlan *et al*., 2002). Upon connecting to CPP-Tutor website, the student's browser displays the working environment for CPP-Tutor. An appropriate skill-level problem is selected or the problem that last attempted is presented to the student. At any time the student can request a new problem by pressing New problem button. The student types in his solution in the Solution window. Once the student finish answering a problem, he can press the Compile button.

After pressing the Compile button, the student code is sent to the C++ compiler. If the student solution pass the compile stage, the C++ engine runs the student code and the output is displayed in the output window; however, if the compile stage fails, a pattern matching procedure is invoked to calculate the edit distance between the student solution and the actual solution, a modified solution of the student code is suggested and information gathered is sent to the expert module for proper feedback.

When the student become more experienced programmer, he or she can skip the tutorial on a specific problem by pressing the run button instead of compile button. The result of the C++ engine is displayed in output window directly.

The student, at any time, may explicitly request from CPP-Tutor to view the solution, Exit from the current problem and select a new one and view his performance


CPP-Tutor an Intelligent Tutoring System
CPP-Tutor
An Intelligent Tutoring System for
Student Learning to Program in C++
Problem description
New Problem
Student Solution
CLS
Compile
Run
Solution
Stat
Exit
Output Window

Problem:
Write a program in C++ which find and print the
cubic value for each integer between -5 and +5

```
#include<iostream.h>
void main(
{
 Int c, j=-5
 while(j>=5) {
   c=j*j*j;
```

```
CPP-Tutor: "It seems something is missing in line 2.
It might be ')' between '(' and ')' Do you agree?"
Student: Yes
```

Fig. 4: CPP-tutor user interface

based on statistics including problems attempted, problems solved, number of attempts on a problem and problem difficulty. The CPP-Tutor user interface is shown in Fig. 4.

CONCLUSIONS

In this study, we have presented recent developments related to the C++ Intelligent Tutoring System, which is based on sound theories, pattern recognition techniques and error-correction strategies. The ultimate goal of the pattern recognition techniques, in CPP-Tutor is to understand the 'intention' of the student by carefully analyzing the student's code and to communicate this to the Expert module to effectively tutor the student through C++ programming problems. This research is significant since it has the potential to be applied to many programming courses at the University level. Furthermore, C++ is an extremely popular first programming language everywhere.

FUTURE WORK

CPP-Tutor has not yet been fully evaluated as to its effectiveness as a tutoring tool. A full evaluation is planned for early 2008, to be taken with an introductory C++ programming class. Although supporting the entire C++ domain is beyond the immediate scope of this project, the system can be incrementally improved through developing more constraints, therefore extending the domain coverage. In addition, the problem set can also be increased, by either working inside a template to develop more complex problems, or developing new templates that focus on different problem goals.

REFERENCES

Abu Naser, S.S. and O. Sulisel, 2000. The effect of using computer aided instruction on performance of 10th grade biology in Gaza. J. Coll. Educ., 4: 9-37.

Abu Naser, S.S., 2001. A comparative study between animated intelligent tutoring systems AITS and video-based intelligent tutoring systems VITS. Al-Aqsa Univ. J., 5: 72-96.

Abu Naser, S.S., 2006. Intelligent Tutoring System (ITS) for teaching database to sophomore students in Gaza and its effect on their performance. Inform. Technol. J., 5: 916-922.

Abu Naser, S.S., 2008. Developing visualization tool for teaching AI searching algorithms. Inform. Technol. J., 7: 350-355.

Alexandrescu, A., 2001. Modern C++ Design: Generic Programming and Design Patterns Applied (C++ In-Depth Series). 1st Edn. Addison Wesley, New York.

Anderson, J.R. and E. Skwarecki, 1986. The automated tutoring of introductory computer programming. Commun. ACM, 29: 842-849.

Anderson, J.R., A.T. Corbett, K.R. Koedinger and R. Pelletier, 1995. Cognitive tutors: Lessons learned. J. Learning Sci., 4: 167-207.

Brusilovsky, P., E. Schwarz and G. Weber, 1996. ELM--ART: An intelligent tutoring system on World Wide Web. ELM--ART: An intelligent tutoring system on World Wide Web. Lecture Notes Comput. Sci., 1086: 261-269.

Conlan, O., C. Hockemeyer, V. Wade, D. Albert and M. Gargan, 2002. An architecture for integrating adaptive hypermedia service with open learning environments. Proceedings of ED-MEDIA 2002, Vol. 1 of World Conference on Educational Multimedia, June 24-29, Hypermedia and Telecommunications, pp: 344-350.

Graesser, A.C. and N.K. Person, 2001. Teaching tactics and dialog in auto tutor. Int. Artificial Intel. Educ., 12: 12-23.

Hong, J., 2004. Guided programming and automated error analysis in an intelligent prolog tutor. Int. J. Hum. Comput. Stud., 61: 505-534.

Johnson, W.L. and S.E. Proust, 1985. Knowledge-based program understanding. IEEE Trans. Software Eng., 11: 267-275.

Koedinger, K.R., 2001. Cognitive Tutors. In: Smart Machines in Education, Forbus, K.D. and P.J. Feltovich (Eds.). MIT Press, Cambridge, MA, pp: 145-167.

Kumar, A.N., 2002. Model-based reasoning for domain modeling in a web-based intelligent tutoring system to help students learn to debug C++ programs. Intel. Tutoring Syst., 2363: 792-801.

Mark, M. and J.E. Greer, 1995. The VCR tutor: Effective instruction for device operation. J. Learn Sci., 4: 209-246.

Qiu, L. and C. Riesbeck, 2004. An incremental model for developing computer-based learning environments for problem-based learning. Proceedings of IEEE International Conference on Advance Learning Technologies, 22 Aug, pp: 908-916.

Shute, V.J. and R. Glaser, 1990. A large-scale evaluation of an intelligent discovery world: Smithtown. Interactive Learn. Environ., 1: 51-57.

Silverman, B.G., 1992. Survey of expert critiquing systems: Practical and theoretical frontiers. Commun. ACM, 35: 106-127.

Soloway, E., E. Rubin, B. Woolf, W.L. Johnson and J. Bonar, 1983. Meno ii: An ai-based programming tutor. J. Computer-Based Instruct., 10: 20-34.

Song, J.S., S.H. Hahn, K.Y. Tak and J.H. Kim, 1997. An intelligent tutoring system for introductory C language course. Comput. Educ., 28: 93-102.

Sykes, E.R. and F. Franek, 2003. A prototype for an intelligent tutoring system for students learning to program in Java. Proceedings of the IASTED International Conference on Computers and Advanced Technology in Education, 30 June- 2 July, Rhodes, Greece, pp: 78-83.

Weber, G. and A. Möllenberg, 1995. A. ELM-programming-environment: A tutoring system for lisp beginners. In: Cognition and Computer Programming, Ablex (Ed.). Norwood, New Jersey, pp: 373-408.

Weber, G. and M. Specht, 1997. User modeling and adaptive navigation support in www based tutoring systems. In: 6th International Conference on User Modeling, June 1997, Sardinia, Italy, pp: 289-300.

Woolf, B.P., J. Beck, C. Eliot and M. Stern, 2001. Growth and Maturity of Intelligent Tutoring Systems. In: Smart Machines in Education, Forbus, K.D. and P.J. Feltovich (Eds.). MIT Press, Cambridge, UK., pp: 100-144.