

Learning Computer Programming using e-learning as a tool.

A Thesis

Submitted to the Department of Computer Science and Engineering

of

BRAC University

by

Asharf Alam

Student ID: 03101011

Md. Saddam Hossain

Student ID: 03101054

In Partial Fulfillment of the

Requirements for the Degree

Of

Bachelor of Science in Computer Science & Engineering

May 2008

## **DECLARATION**

We hereby declare that this thesis is based on the result which is found by ourselves. Materials of work found by other researchers are mentioned by references. This thesis, neither in whole nor in part, has been previously submitted for any degree.

Signature of  
Supervisor

Signature of  
Author

## **Acknowledgement**

I wish to express my heartiest gratitude, sincere appreciation and indebtedness to my, honorable thesis Supervisor Zillur Rahman and co-supervisor Risat Mahmud Pathan, for their guidance, encouragement, support and the constructive suggestions throughout the completion of this report. We would like to thank all peoples who helped us to prepare this final report.

Finally I pay my deepest respect to ALLAH the beneficial and the merciful and thank him for the successful completion of the report.

## **Abstract:**

This thesis report seeks to design idea for e-learning programming tools. An intelligent education system could change one's life towards a better future. In this software design, we used Visual Basic programming language for making demo version. The main purpose of the software is to give basic ideas about the programming language for the new programmer. This software will create questions for the user and will evaluate those answers. There are different complexity levels such as beginner, advanced, and professional. Beginner users are those who are completely new in the computer programming area. There will be different sets of questions for the different types of users. From the user answer, we will evaluate the user's level. When they complete all of the level successfully then we can say that he/she has a better idea about programming.

## **Objective:**

The role behind this study is "How effectively we can use e-learning tools for learning programming?" which is of low cost but in an effective and efficient way. We are trying to teach programming language to the people with the help of technology. Our main objective is "Teach programming with the help of e-learning as a tool in an effective way". Here we give some basic implementation idea that will help the user for learning programming.

## Table Of Contents

1. Introduction	8
2. Literature Review	9
3. Motivation	11
4. Solution	12
4.1. Fundamental of computer	13
4.2. Introducing to programming language	14
4.3. Introduction to C++	15
4.4. Fundamentals of Algorithm	15
4.5. Basic Data type	16
4.6. Declaration with initialization	19
4.7. User interface	19
4.8. Compiling	24
4.9. Variables	25
4.9.1. Parts of variables	26
4.9.2. Types of variables	26
4.10. Storage	27
4.11. Identifier	27
4.12. CIN and COUT	27
4.13. Input output statement	29
4.14. C++ I/O class library	30
4.15. Operator	30
4.15.1. Arithmetic Operator	31
4.15.2. Relational Operator	31
4.15.3. Bitwise Operator	31
4.15.4. Logical Operator	32
5. Programming syntax and semantics	39
6. Step that we have followed	41
7. Future Work	42
8. Conclusion	42
7. References	43

## List of Figures

1.	Fig. 4.1.	New User Interface	12
2.	Fig. 4.2.	Existing User Interface	13
3.	Fig. 4.3.	User Question Interface	14
4.	Fig. 4.4.	User Question Interfaces	15
5.	Fig. 4.5.	Basic Data Type	17
6.	Fig. 4.6.	Declaration with Initialization	17
7.	Fig. 4.7.	Declaration with Initialization	18
8.	Fig. 4.8.	Declaration with Initialization	18
9.	Fig. 4.9.	Sum of Integers	19
10.	Fig. 4.10.	Sum of Integers	20
11.	Fig. 4.11.	User Question Interfaces	21
12.	Fig. 4.12.	User Question Interfaces	23
13.	Fig. 4.13.	User Question Interfaces	24
14.	Fig. 4.14.	User Question Interfaces	25
15.	Fig. 4.15.	User Question Interfaces	25
16.	Fig. 4.16.	User Question Interfaces	26
17.	Fig. 4.17.	User Question Interfaces	26
18.	Fig. 4.18.	Identifier	27
19.	Fig. 4.19.	Cin/Cout	28
20.	Fig. 4.20.	I/O statement	29
21.	Fig. 4.21.	C++ Class Library	30
22.	Fig. 4.22.	If/else statement	35
23.	Fig.4.23.	Evaluation	36
24.	Fig.4.24.	Evaluation	36
25.	Fig.4.25.	For structure	38
26.	Fig.4.26.	Evaluation	38
27.	Fig.4.27.	Evaluation	39

## List of Coding

1. Declaration with initialization	19
2. Sum of two integers	22
3. Cin and Cout	28
4. Logical operator	32
5. If else statement	34
6. For structure	37

# **1. INTRODUCTION**

In our Artificial intelligence base topic we are trying to outlines the results of a small-scale action research-study that investigated how well various instructional strategies translate to a text-based programming learning environment and which are most effective for facilitating higher levels of programming learning. We are trying to introduce very basic part of programming language to those programming learner who are in primary level. We introduced them about the definition of computer, definition of programming language, gave the idea about the logical operator, arithmetic and logical operation and so on. Programming is the backbone of a computer science student. It is very important to have a better idea about programming for future. But unfortunately in our country most of the computer science students are frightened about programming. As a result they are going to change their job place. Now a days computer science background student join as a marketing executive or sales executive to the other company for the lacking of idea in programming field. There are so many software firms in our country. But their prerequisite is, the candidate should have a better idea about programming. But for avoidance of programming, so many students do not bother to join in a software firm. That is why we are trying to give a brief idea on how can we improve the programming level of our computer science students so that they can compete with other for better position in job market. These are the very basic parts of our thesis.



## **2. Literature Review**

For e-learning there are various kinds of software. We are going to implement software which will be very easy to understand and students feel comfortable to work with this software. This literature review is a consideration of the issues associated with the infrastructural aspects and the need to associate the usefulness of technology to enhance the learning experience. This technological path will potentially enhance the learning process, not replace the lecturer or tutor. For lecturers and students, the implications of eLearning are extensive. It is widely acknowledged that implementation of eLearning leads to a fundamental shift in learning styles; however research into the effects of this shift is inconclusive. Singh and Priola [14] summaries a number of opposing views. Firstly, Knight [8] proposes that eLearning will benefit students who are used to being 'spoon fed' on the basis that students can no longer be passive about their learning. This view is endorsed by Hawkes and Cambre [5] who claim that in order to gain results, students must take responsibility for their own learning. Secondly, and in contrast to Knight, the views of Kershaw are noted. Kershaw [7] proposes that students will not automatically become conscientious, self motivated individuals and that success in fact depends on the level of interaction between students and lecturers that is required to stimulate good results. Based on the lack of conclusive evidence relating to the effects of a change in learning style, it seems appropriate to assume that not all students respond well to an eLearning environment. Cooper [1] points out that independent learner have the potential to be successful in distance education, however those lacking in the skills to study independently will not react well in a virtual environment. Under such circumstances, institutions implementing eLearning must be aware that students will react differently to the changing paradigm of learning and rather than implement changes across the board, should aim to offer courses tailored specifically towards the different learning styles.

The use of technology in education and training is transforming the way that people learn in today's academic and corporate settings. According to the 1999

Training Industry Report, technology training budgets in the corporate setting increased 13% from 1998 to 1999.

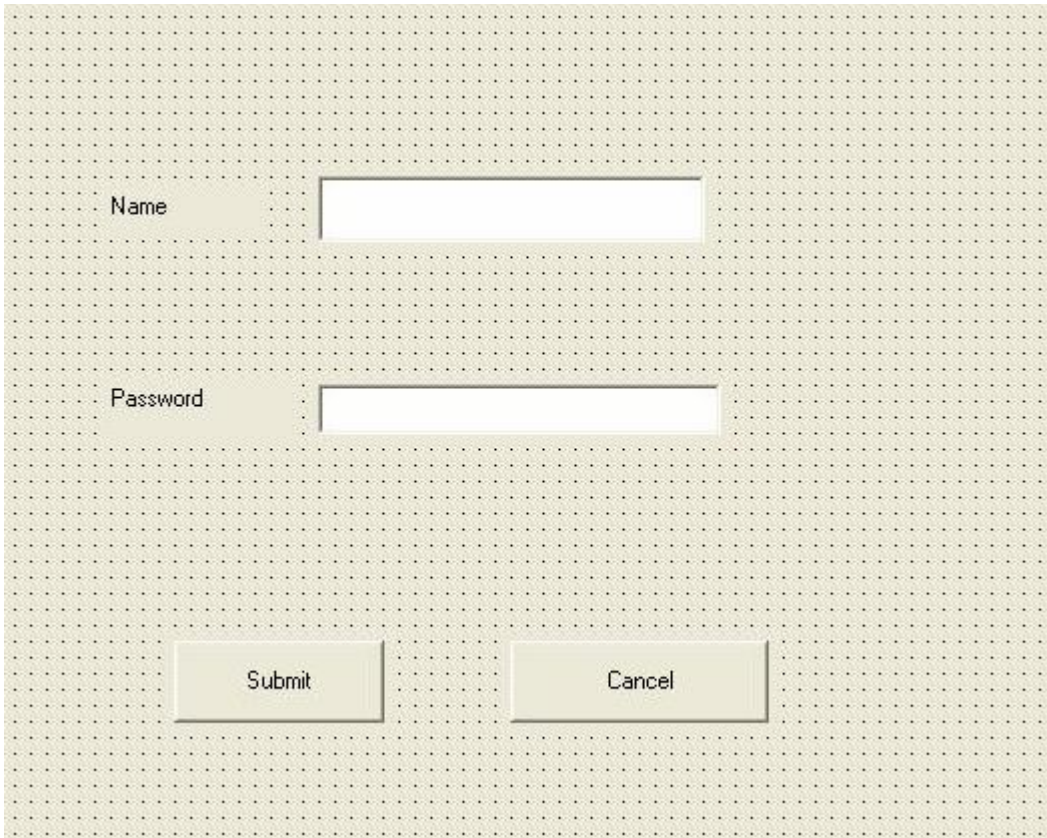
According to McFadzean [9] traditional teaching and learning skills need to change in order to get maximum benefit from virtual learning. While technology alone might not be the answer to all of the university's problems, according to Daniel [2], it certainly can play a key role. According to Redfern & Naughton [13] the benefits of utilizing technology, particularly for developing online collaborative activities are well documented. According to O'Donoghue & Singh [11] technology is a powerful medium particularly for part time work based students who find erratic attendance requirements and study difficult. The rapid growth in eLearning, experienced particularly during the 1990s, has overcome many of the barriers to Higher Education told by National Committee of Enquiry into Higher Education [10], providing traditional universities with an opportunity to meet the changing worldwide demand for education. According to Goddar [4] the demand for higher education is expanding exponentially throughout the world and by 2025 as many as 150 million people will be seeking Higher Education. According to Katz [6] this increase in demand is widely attributed to the changing culture of employment, where a job for life is no longer the norm, and to the advent of the so-called 'knowledge-driven society. According to Davis [3] society requires higher levels of skills and qualifications to fill the same 'worthwhile' jobs, and individuals see education as a status provider told by Pritchard & Jones [12]. Volery and Lord [15] point to the capacity constraints and resource limitations that can be overcome through the implementation of eLearning, creating a new opportunity to satisfy this growing demand.

### **3. Motivation**

In our thesis we give implementation idea of software which is used for e-learning programming language. This software basically used for those who are completely new in programming. In the past there was a lot of work on e-learning process. But in the undergraduate level there were only few tasks on this learning tool. And in this e-learning programming software using fundamental concept for learning programming which is most beneficial for the new level programmer. That is why we are trying to give idea of implementation of a software which is helpful for the user to learn programming. This is a vast area to cover and that is why we have done some parts of it. Such as we have covered initially about the basic knowledge about computer, fundamentals of programming language and it's purpose, brief description about variables, knowledge about compiling, information about input and output statement and how can we give input to the computer and get the output from the computer, definition of different types of loops etc. Then we define grammar for a computer program and also define the mandatory and optional part of a program. In our thesis we are trying to develop software which has covered till for loop. It gives the basic idea about programming and how to write a program correctly and efficiently.

## 4. Solution

This is a user interface for login. A fresh user has to give his/her name as login id and a password for creating an account. After successful creation there is a notification message that a new account has been created with login information. There is a database where all of the information is stored.



The image shows a user interface for creating a new account. It consists of a light beige background with a fine grid pattern. There are two input fields: "Name" and "Password", each with a corresponding text label to its left. Below the input fields are two buttons: "Submit" and "Cancel".

Fig.4.1: This is the new user interface. They give their name and password for creating a new account and submit it. There will be a database where we store all the data about the user.

The image shows a login form on a light gray background with a fine grid pattern. On the left side, there are two labels: 'User ID' and 'Password', each in a light gray rectangular box. To the right of each label is a white rectangular input field with a thin gray border. Below these two input fields is a single light gray rectangular button with the text 'sign In' centered inside it.

Fig.4.2. This is allocated for the existing user who has already an account. They give their name and password and the software will check this information in the database. If it matches then the user can log in. Otherwise an error message will occur.

## **4.1 Fundamentals of Computer**

We are trying to design software which will help to learn programming. This programming language runs on computer. A computer is an electronic device that accepts information and manipulates it for some results based on sequence of instructions given by software on how the data is to be processed. There are two basic parts of a computer: Hardware and Software. Hardware cannot run by itself, it can run through software. Software is the information that the computer uses to get the job done.

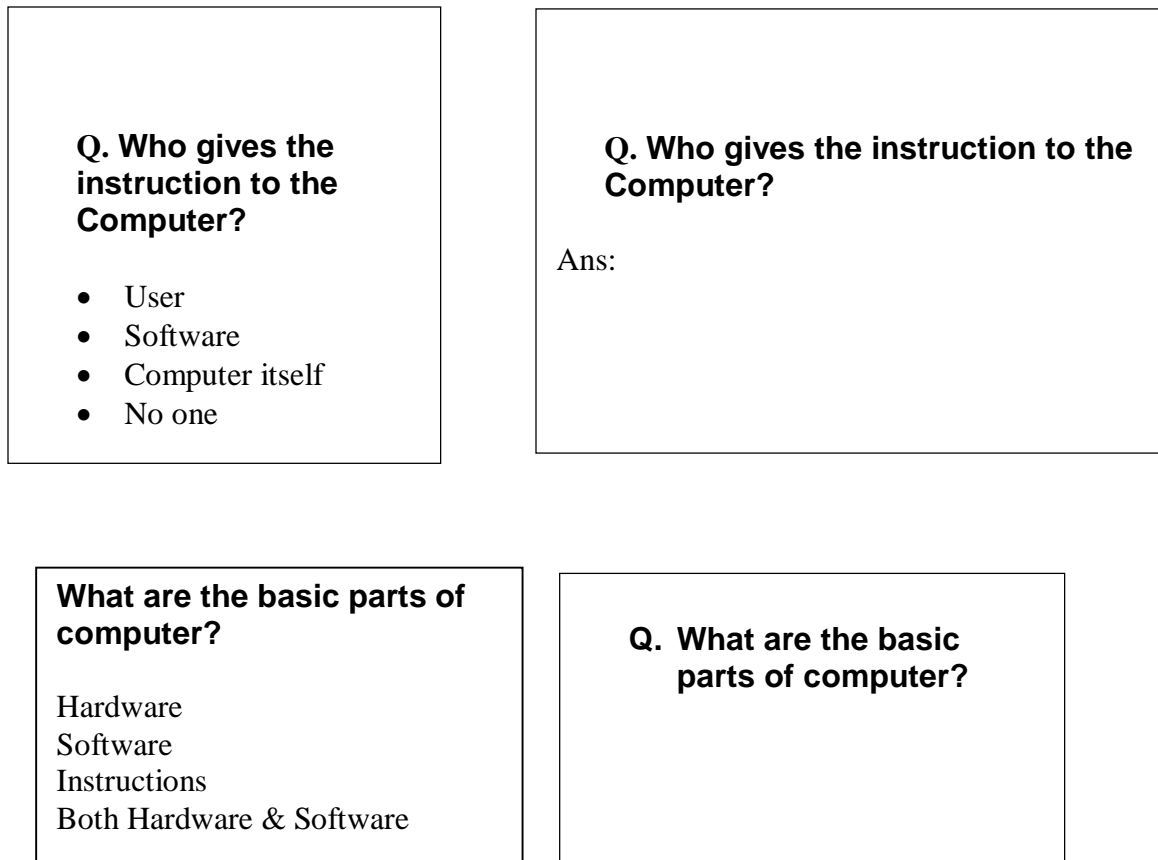


Fig.4.3. Users need to answer these questions.

## **4.2 Introducing to Programming Language**

There are different levels of programming language such as high level or low level. A computer only knows machine language. So when we write a program using a language, this is only some instructions to the computer. It describes how it should work, what should be done.

Although computer science and engineering has developed enormously it is surprising to know that there is no general agreement on the meaning of the term “programming languages”. The general understanding is that it is a collection of rules that instruct computer to perform specific tasks.

There are three levels of languages: High-level language, Assembly language and Machine language. Programming language usually refers to high-level

languages such as QBASIC, C, C++, COBOL, FORTRAN, Ada and Pascal. All these languages have unique keywords. Computer cannot understand high level languages because it is more similar to human languages. Assembly languages are similar to machine languages. In case of assembly language programmer can use name instead of numbers. Computer can actually understand machine languages. Machine languages consist of binary numbers. It is almost impossible for human to understand machine languages.

<p><b>Q. What is programming language?</b></p> <ul style="list-style-type: none"><li>• A set of instructions</li><li>• A Software</li><li>• Computer Component</li><li>• None of the above</li></ul>	<p><b>Q. What is programming language?</b></p> <p>Ans:</p>
--	--

Fig: 4.4 User need to answer these questions

### **4.3. Introduction to C++**

C++, is an extension of C, was developed by Bjarne Stroustrup in the early 1980s at Bell Laboratories. C++ provides a number of features that “spruce up” the C language, but more importantly, it provides capabilities for object-oriented programming.

### **4.4. Fundamentals of Algorithm**

Any computing problem can be solved by executing a sequential order of actions that will help to reach desired result. This structured procedure is called

Algorithm. Algorithm helps us to minimize our task with its efficiency. An algorithm is efficient when its cost is low but the output is desired.

Real-world Problems □ Possible Solution □ Steps of the solution

- Graphical/Textual Representation (Algorithm) of the steps of the solution
- Using programming language=Implementation of the algorithm=CODING

Sequence of computational steps that transform the *input* into the *output*

tool for solving a well-specified *computational problem*

*program* is the expression of *algorithm* in a *programming language*

## **4.5. Basic Data Types**

Data types are used for determining what type of data a program contains. It is important to verify the data types.

There are few basic data types in programming. Those are

- int
- float
- double
- bool
- char



## Basic Data Type

<b>5</b>	<b>int</b>
<b>5.0</b>	<b>float</b>
<b>c</b>	<b>char</b>

Fig. 4.5 is allocated for the basic data types. Users have to identify the accurate data type.

## Evaluation

<b>a=10</b>	<b>b=5</b>
<input type="text"/>	<b>a, b;</b>

Fig 4.6 is allocated for the user evaluation form.

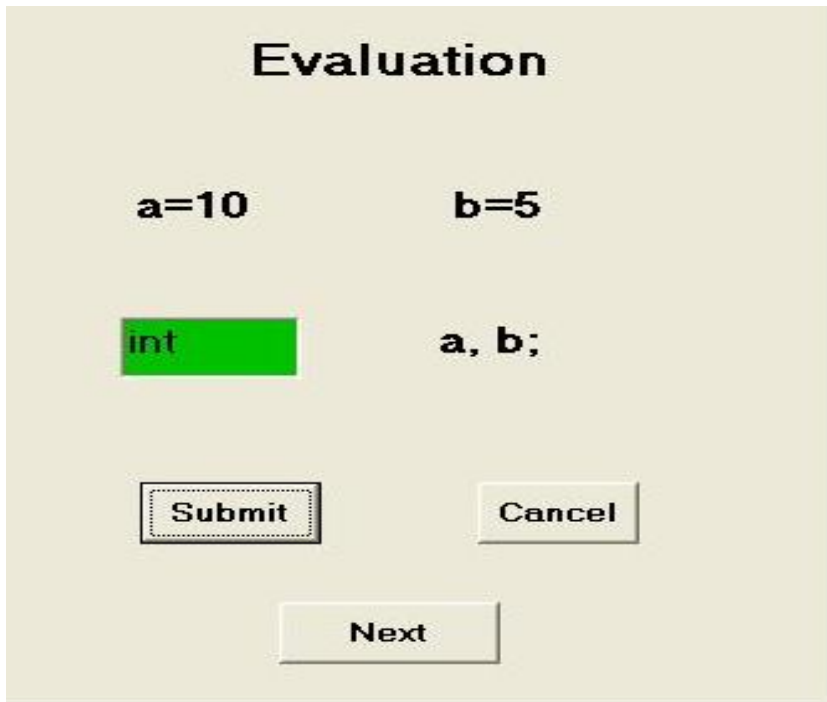


Fig 4.7:- This is allocated for the evaluation form, if the answer is right the box will be green.

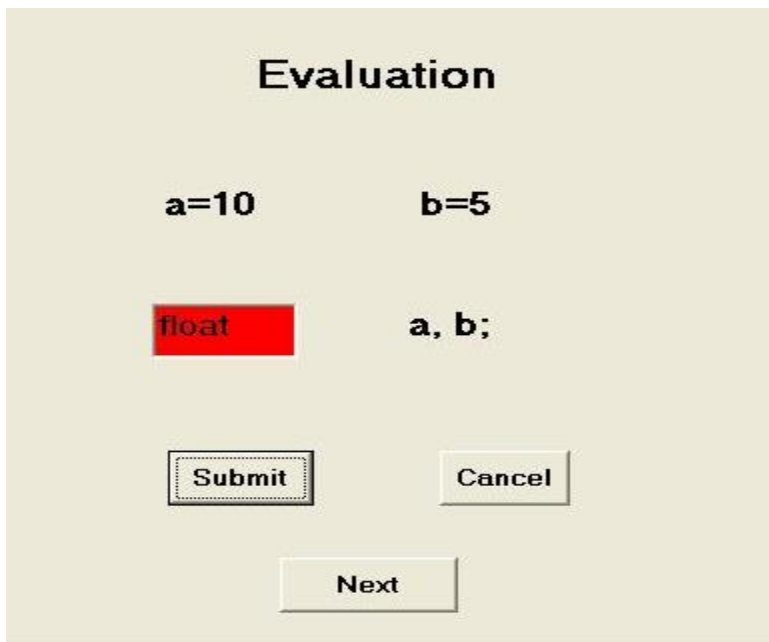


Fig 4.8:- This is allocated for the evaluation form, if the given answer is wrong the box will be red.

## 4.6. Declaration with Initialization & without Initialization:

In any function there is a need for declaration. Otherwise the function is not complete and it does not work properly. Declaration defines a specific path for the function. For example: here is a declaration with initialization & without initialization:

```
int i = int();
```

Here we have declared an integer type variable i and its initial value can be anything.

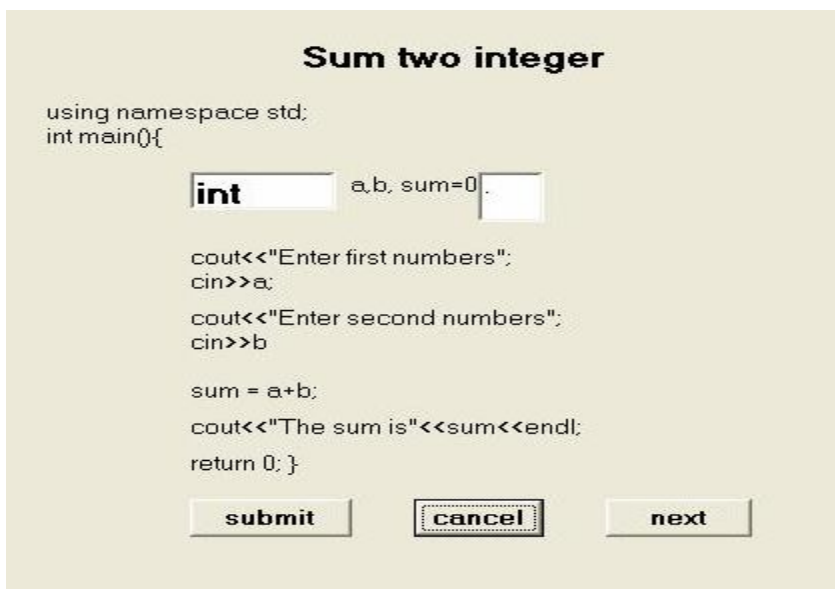
```
int*countptr, count;
```

```
int y = 5; (pointer operator)
```

```
Y ptr      y
```

Here we have declared a variable count using a pointer. And counter will increase as time progress.

## 4.7. User Interface



The screenshot shows a C++ program titled "Sum two integer". The code is as follows:

```
using namespace std;
int main(){
    int a,b, sum=0;
    cout<<"Enter first numbers";
    cin>>a;
    cout<<"Enter second numbers";
    cin>>b;
    sum = a+b;
    cout<<"The sum is"<<sum<<endl;
    return 0; }
```

The user interface includes three input fields: one for the variable 'int', and two for 'a,b, sum=0'. Below the code, there are three buttons: 'submit', 'cancel', and 'next'.

Fig. 4.9 This is allocated for the variables with initialization with mandatory declarations.

### Sum two integer

```
using namespace std;
int main(){
    int a,b, sum=0;

    cout<<"Enter first numbers";
    cin>>a;

    cout<<"Enter second numbers";
    cin>>b;

    sum = a+b;

    cout<<"The sum is"<<sum<<endl;
    return 0; }
```

Fig. 4.10 is allocated for checking the answer; if the learner gives the wrong answer then the block will be red color.

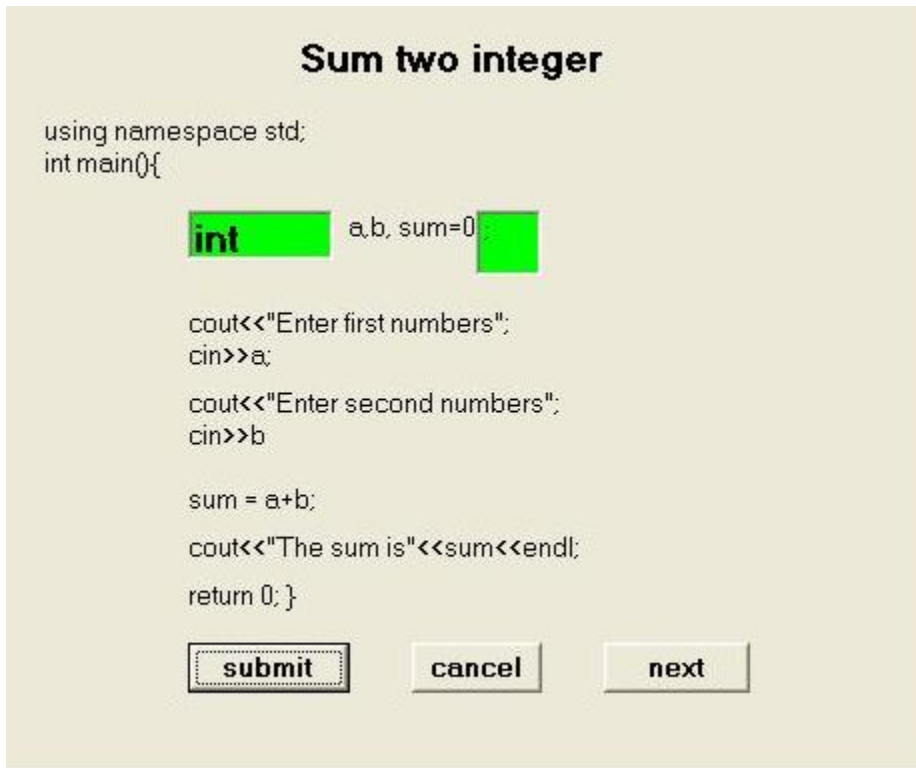


Fig 4.11 shows that if the learner gives the right answer then the block will be green color.

Incase of adding two numbers we can go through the following pseudo code.

Step 1: At first we have declared two integer type variables a and b and also initialize the sum=0, a=0&b=0.

Step 2: We are taking inputs from the user for those variables a and b.

Step 3: Then we add those inputs and store the result in sum.

Incase of adding two numbers we can go through the following pseudo code.

Step 1: At first we declared two integers type variable a and b and also initialize the sum=0, a=0&b=0.

Step 2: We are taking inputs from the user for those variables a and b.

Step 3: Then we add those inputs and store the result in sum.

This is an example for sum of two integers and after that there is a demonstration for optional and mandatory part:

```
#include <iostream>          //[Library function]
using namespace std;        //[For cin and cout]

int main()                  //[Return type]
{
int a,b,sum=0;

cout<<"Enter first number: ";
cin>> a;

cout<<"Enter second number: ";
cin>>b;

sum=a+b;

cout<<"The sum is: "<< sum <<endl;

return 0;

}
```

### **Sum of two integers:**

```
using namespace std;
int main()
{
```

```
 a, b, sum=0 
```

```
cout<<"Enter first number: ";  
cin>> a;  
cout<<"Enter second number: ";  
cin>>b;  
sum=a+b;  
cout<<"The sum is: "<< sum <<endl;  
return 0;
```

Here 'int' and ";" are missing. Therefore learner should fill up the block which consist this "?". This type of problem is allocated for primary level learner.

**Sum two integer**

```
using namespace std;  
int main(){  
 a,b, sum=0   
  
cout<<"Enter first numbers";  
cin>>a;  
  
cout<<"Enter second numbers";  
cin>>b  
  
sum = a+b;  
cout<<"The sum is"<<sum<<endl;  
return 0; }
```

Fig.4.12. In the above figure: This is a sample question.

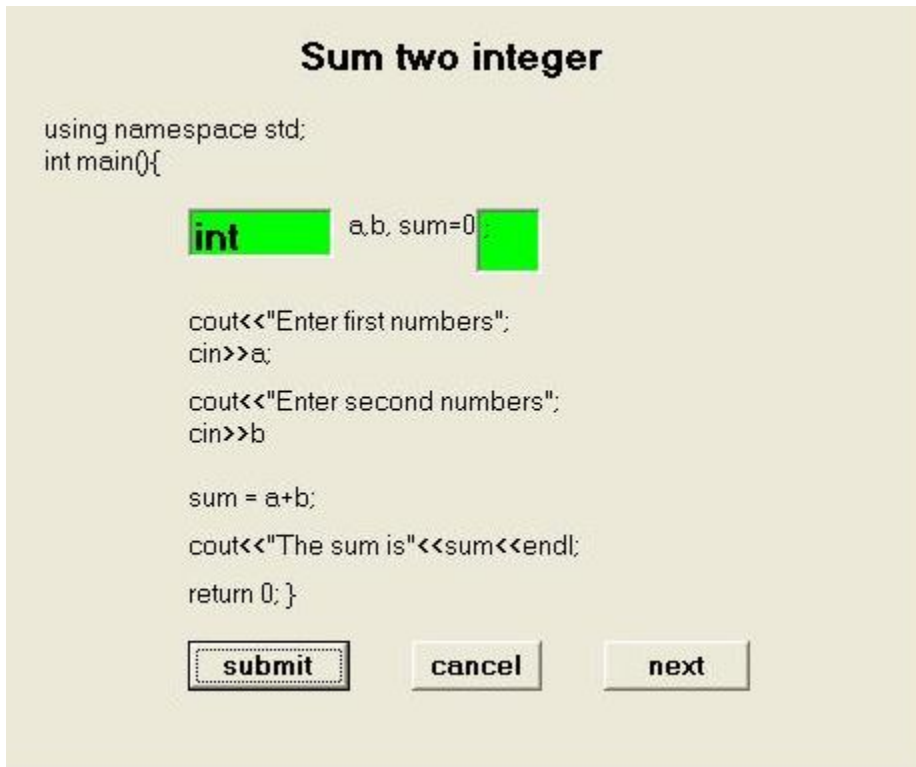


Fig.4.13: This is for evaluating right or wrong answer. If the answer is right then it will be green color.

## 4.8. Compiling

Compiling is the method of transforming a program written in a high-level programming language from source code into object code. Programmers write programs in source code. Source code must be modified before it can be used as an executable program. The first step is to pass the source code through a compiler, which translates the high-level language instructions into an object code. The final step in producing an executable program is to pass the object code through a linker. The linker combines modules and gives real values to all symbolic addresses, to produce a machine code.



<p><b>Q. What is the process to compile a program?</b></p> <p>Ans:</p>
--

Fig. 4.14: shows that users need to write the text for the given question.

### 4.9. Variables

In our program there are different types of objects which are useful for our task. For our convenient use we give a name as identifies for each of them. This object is called variable. Variable named because they are changing for each object.

As example:- `int interger1; int interger2; int sum;` word `interger1, interger2` and `sum` are the names of variables. A variable is a location in the computer's memory where a value can be stored for use by a program. This declaration specifies that the variables `integer1, integer2, and sum` and data type are `int`, which means that these variables will hold integer values such as `7,-10,100`. All variables must be declared with a name and a data type before they can be used in a program. We can also declare `float and double char` in this way.

<p><b>Q. One variable for multiple values?</b></p> <ul style="list-style-type: none"> <li>• Possible</li> <li>• Not possible</li> </ul>	<p>Q. Char variable only can take alphabet or digit.</p> <ul style="list-style-type: none"> <li>• Yes</li> <li>• No</li> <li>• None of the above</li> </ul>	<p>Q. What happen if we use a variable in a program for integer and for a char?</p> <ul style="list-style-type: none"> <li>• Program will crash</li> <li>• Shows error</li> <li>• Recompile</li> <li>• None of the above</li> </ul>
---	---	---

Fig. 4.15 shows that users need to identify the right answer.

### 4.9.1. Parts of variables

There are some parts within a variable. First part indicates the type of variable which may be integer, float, double or character. Second part indicates the variable name and then a statement.

For example: `int a;` this is the declaration of an integer. Here `int` is the data type and `a` is the variable and semicolon is the ending statement, `a` is the integer value so it can store only integer value.

<p>If we mistake to declare a variable for the integer then what will happen? It shows error Display to declare Program will crash. None of the above</p>	<p>Q. If we mistake to give semicolon what happen to the program?</p> <ul style="list-style-type: none"><li>• It shows error</li><li>• Program will run</li><li>• None of the above</li></ul>
---	---

Fig. 4.16 shows that users need to identify the right answer.

### 4.9.2. Types of variables

There are different types of variables. These are integer, float, double, character. There are some certain ranges for these variables. The ranges for each variable are given below:

Int takes the values between -32767 to 32767

Float takes the values between -2147483647 to 2147483647

Char takes the values between -127 to 127

<p>How the number of range is defining? Ans:</p>	<p>What happen if you give the big number than the range? Shows error It takes the given value Prints garbage value None of the above</p>
--	---

Fig. 4.17 shows that users need to identify the right answer as well as write the text for the given question.

## 4.10. Storage

To make the computer system convenient to the user, the operating system provides a uniform, logical view of information storage. Storage means store the data for future use. In a computer there are some specific storage media which performs a vital role for data manipulation, data storage and faster use of data.

Program activation is the process of allocating and initializing static storage for the program. Program invocation is the process of allocating memory.

## 4.11. Identifier

The names of variables, function, and various other user-define items are called identifier. The length of this identifier can vary from one to several characters. The first character must be a letter or an underscore, and subsequent characters must be letters, digit, or underscores.

Valid identifier: count, c\_23, c\_b

Invalid identifier:- 1count, h!the, h...b

<p>Q. What is the user-define item?</p> <p>Ans:</p>	<p>Q. What is identifier?</p> <p>Ans:</p>
---	---

Fig. 4.18: shows a user interface where users need to answer the questions.

## 4.12. CIN and COUT

In programming there is a need for giving something to the program as input and after evaluation that input program returns something which is output.

In C++ language cin and cout used for input and output operations.

For example, `cin>>integer1;` it obtains a value for variable integer one from the user.

And for cout: `cout<<"Enter two integers"`

```
cin>>num1>>num2;
```

Here, cout prints the message "Enter two integers" and cin reads two integers. num1 and num2 are declared at the first part of a program.

Q. Write down the purpose of cout and cin?

Ans:

Q. How can I get inputs from user?

Ans:

Q. How can I print a message on the computer screen?

Ans:

Fig. 4.19: This interface is used for user evaluation.

Example for declaring variables:

```
int x;
```

```
int y;
```

This is how we declare variables. Here, two integer type variables x and y.

Example: `int y =5;`

Here, we define a value for the int type variable y, which is 5.

Cout:

- cout: cout is used for printing.

```
cout<< *yptr << endl;
```

Here, it will print the pointed value of y.

Example: `cout<<"Welcome to our THESIS"`

It will print the sentence enclosed by (" ")

- `cout<< 5;` (Directly print a value 5)
- `cout<< a;` ( It will print the value of "a" variable )
- `cout<< a+b;` ( It will print the summation value of variables a and b)

### **4.13. Input/Output statement:-**

C++ I/O occurs in streams, which are sequence of bytes. In input operations, the bytes flow from a device to main memory. In output operations, bytes flow from main memory to device. This byte could represent characters, raw data, graphic images, digital speech or videos etc.

The system I/O mechanisms should transfer bytes from device to memory consistently and reliably. This I/O operation requires careful planning to ensure maximum performance. C++ provides both “low-level” and “high-level” I/O capabilities. Low-level I/O capabilities specify that some number of bytes should be transferred device to memory or memory to device. It provides high speed, high volume transfers but is not particularly convenient. Programmers generally prefer a higher level view of I/O, in which bytes are grouped into meaningful units, such as integers, floating point numbers, characters, strings etc.

<p>Q. I/O statement: How it works?</p> <p>Ans:</p>	<p>Q. What types of I/O capabilities that C++ provides?</p> <p>Ans:</p>
<p>What types are used by the programmers?</p> <p>Ans:</p>	<p>In which format I/O occurs and what do you mean by byte?</p> <p>Ans:</p>

Fig. 4.20 shows user has to give the answers of the above questions.

#### **4.14. C++ I/O class library:**

There is a library for C++ language which includes the I/O classes, I/O headers, the format flags, several data types, the general purpose I/O functions.

There are several data types like the streamsize and streamoff types, the streampos and wstreampos types, the pos-type and off-type. Types the openmode type, io state type, seekdir type.

I/O functions are: bad, clear, eof, exceptions, fail, fill, flags, flush, fstream, ifstream, and ofstream, gcount, get, getline, good, ignore, open, peek etc.

<p>What is the task of a library?</p> <p>Ans:</p>	<p>How it maintains its information?</p> <p>Ans:</p>
<p>Does it important to have a library?</p> <p>Ans:</p>	

Fig. 4.21 Users have to write the answer of all the questions.

#### **4.15. Operator**

Operators are usually used for arithmetical operations like adding, subtracting, division and multiplication.

There are four main classes of operators: arithmetic, relational, logical and bitwise. In addition, there are some special operators for particular tasks.

- Operators are: +, -, \*, /, %
- Decision making: \*, /, +, -

For example:  $10 * 5 + 3 * 5 + 7$

- Step1: 50

- Step2:                    + 15 +7
- Step3:            50 +            22
- Step4:                    72

### **4.15.1. Arithmetic operator**

(Types of operator and their actions)

<b>Operator</b>	<b>Action</b>
-	Subtraction
+	Addition
*	Multiplication
/	Division
%	Modulus
--	Decrement
++	Increment

### **4.15.2. Relational Operators**

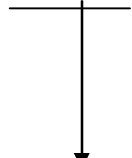
<b>Operator</b>	<b>Action</b>
>	Greater than
>=	Greater than or equal
<	Less than
<=	Less than or equal
==	Equal
!=	Not equal

### **4.15.3. Bitwise Operators**

<b>Operator</b>	<b>Action</b>
&	AND
	OR
^	Exclusive OR (XOR)
~	One's complement (NOT)
>>	Shift right
<<	Shift left

Here, we will see how operator works (with precedence).

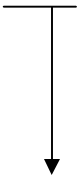
Step1:  $y = 2 * 5 * 5 + 3 * 5 - 7$  [Leftmost multiplication]



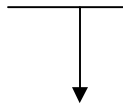
Step2:  $y = 10 * 5 + 3 - 5 * 7$  [Leftmost multiplication]



Step3:  $y = 50 + 3 - 5 * 7$  [Rightmost multiplication]



Step4:  $y = 50 + 3 - 35$  [Leftmost addition]



Step5:  $y = 53 - 35$  [Leftmost Subtraction]



Step6:  $y = 18$  [Last operation - - - place 18 into y]

In the above example it executes multiply operation at first, then addition and at last the subtraction.

#### 4.15.4. Logical Operators

Operator	Action
&&	AND
	OR
!	NOT



## **Logical operator:**

C++ provides logical operators that are used to form more complex condition combining simple conditions. &&(logical and), || (logical Or), ! (Logical Not).

```
If(gender ==1 && age >= 65)
```

```
++ seniorfemales;
```

Here if , && and ; are necessary.

Here we can ignore one condition gender ==1 or age >= 65.

This if structure contains two simple conditions. The gender == 1 is used here to determine whether a person is a female. First we evaluates the left side is true then we evaluates right side, Then if structure then consider the combinational condition. This is && statement so if we found both condition are true then we increase the senior female number.

Consider the condition of || (logical Or). Logical or suppose ensure at some point in a program that either or both of two conditions are true then it will execute the condition.

Ex:-

```
If(semesteraverage >= 90 || finalExam >=90)
```

```
Cout << "Student grade is A";
```

Here if, || and ; are necessary part.

Here we can ignore one condition semesteraverage == 1 or finalexam >= 90.

A student can get A if his semsteraverage is 90 or finalExam mark is 90. He/She can also get A if it satisfies both of the condition;

## **For ! (logical Not):**

C++ provides the! (logical Not) operator to enable a programmer to “reverse” the meaning of a condition; Logical Not operand takes only single condition;

Ex:- if(!(grade == sentinelValue))

```
cout<<"The next grade is" <<grade;
```

Here if (,== and ;) condition are mandatory but we can ignore ! this part;

Here if grade is not equal to sentinelvalue then it will execute the program or not;

If selection structure:-

Example:-

If student's grade is greater than or equal to 60

Print "passed".

**This if selection statement will be this type:**

```
if(grade>= 60)
```

```
cout<<"Passed";
```

Here if ,; and >= part are mandatory.

If/else selection structure:-

If selection structure will be perform only when the condition is true, otherwise the action will be skipped.

Example:-

If student's grade is greater than or equal to 60

Print "passed".

Else

Print"failed".

**If/else selection statement will be this type:**

```
If(grade >= 60)
```

```
cout<<"passed";
```

Else

```
cout<<"Failed";
```

Here if , >= and ; part are mandatory.

But if we ignore else it will be execute but it never fill up the condition failed;

This statement can be written as this way.

```
cout<<(grade >= 60 ? "passed" : "Failed");
```

Here ?, >= and ; are mandatory;

Here, passed condition will be print if the grade  $\geq 60$  is true otherwise failed condition will be print.

Example:-

```
If(grade  $\geq$  60)
    cout<<"Passed";
Else
{
    cout<<"failed";
    cout<<"You must take this course again";
}
```

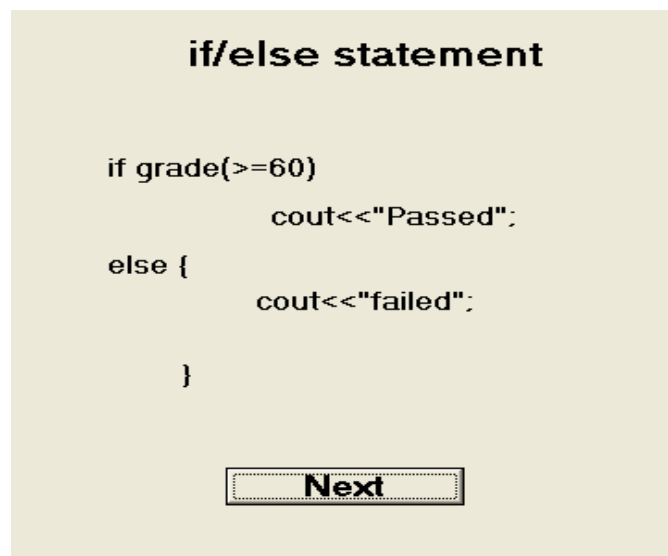


Fig 4.22: The above program will be shown this way.

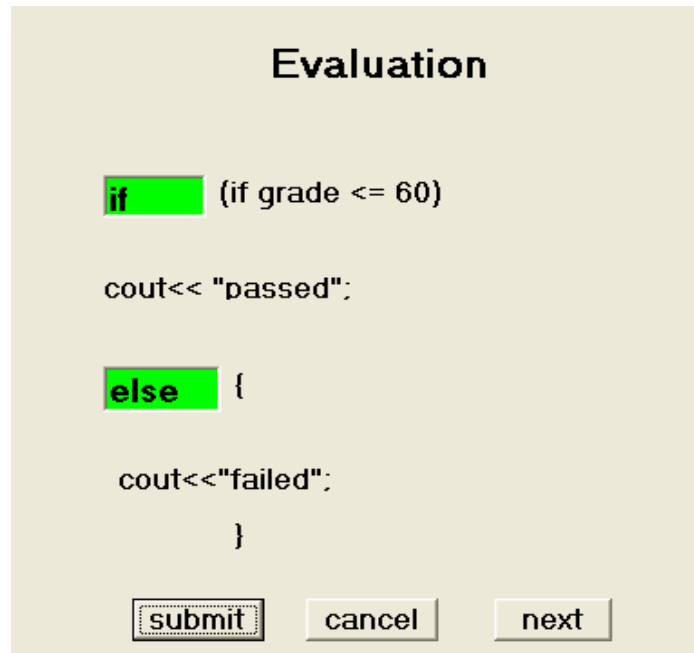


Fig 23: The green box used as a input box and green indicates that the input is correct.

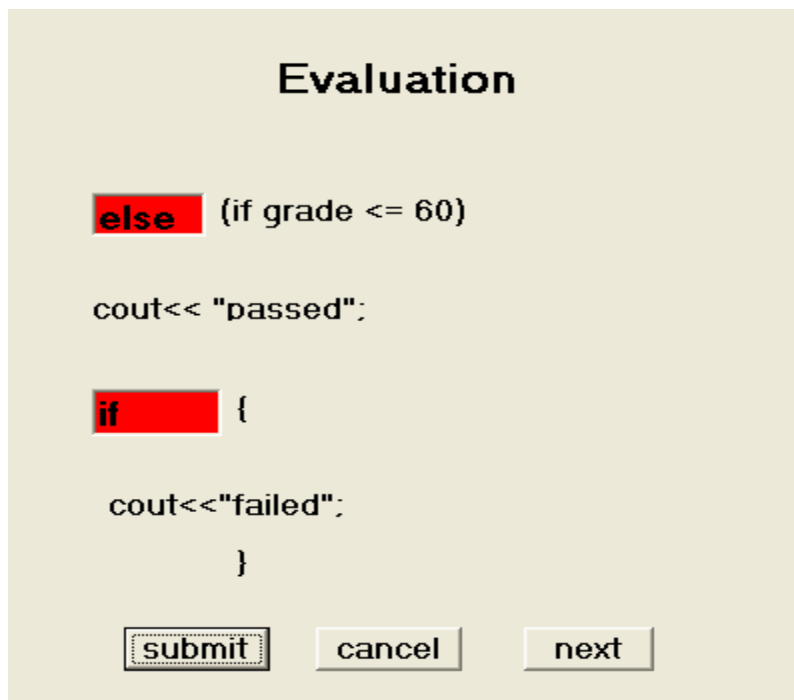


Fig 24: The red box indicates that the input is wrong.

If the grade is less than 60 then this program will be print

Failed

You must take this course again.

If the else statement is without the braces the

cout<<"you must take this course again";

It will be outside of the program and only the

cout<<"failed " statement will be print;

### **While repetition structure:-**

```
int product = 2;
```

```
while(product <= 1000)
```

```
{
```

```
Product = 2*product;
```

```
cout<<product;
```

```
}
```

Here while, <= and ; are mandatory part but we can ignore {} for execute the program.

This product will print 4,8,16.....512, when this program will get 1024 it will be outside of the while condition and it will be execute the next statement of the while condition.

For repetition structure:-

```
int main()
```

```
{
```

```
    for(int counter = 1; counter <=10; counter++)
```

```
        cout<<counter<<endl;
```

```
    return 0;
```

```
}
```

### For structure

```
int main() {  
    for(int counter =1; counter <=10; counter++)  
    {  
        cout<<counter<<endl;  
    }  
    return 0;  
}
```

Figure 4.25: This program will be shown this way.

In this program learner have write a program which print the value from 1-10. Therefore they have to give the proper logic to print this.

### Evaluation

```
for ( int counter = 1; counter<=10; counter++ )  
{  
    cout<<counter<<endl;  
}
```

Fig 4.26: The interface will be these types if all input is correct.

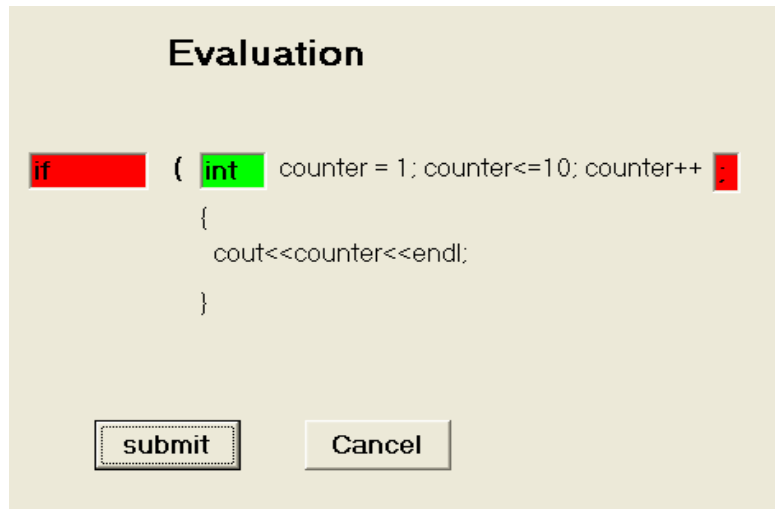


Figure 4.27: Some blocks are red and some are green.

Blocks which hold red colors indicate that the learner gives the wrong answers and the green colors indicate the right answers.

Here counter <= 10 condition is the necessary part.

This program will print 1 to 10;

## **5. Programming syntax and semantics**

In a program there are different parts. It includes program itself, declaration, identifier, parameters, functions, statements etc. In the following part, we determine the grammar for our program.

1. Program -> declaration-list must.
2. declaration list can be declared or null.
3. There are two types of declaration: variable declaration and function declaration.
  - a. variable declaration must have type-specifier id.
  - b. type specifier can be int, float, string, bool.
4. function specifier must have type-specifier ID(params) and compound statement.

- a. Param list must be void.
  - b. Param must have type specifier id or compound statement.
    - i. Compound statement must have local declaration statement list.
      - 1. local declaration must have local declaration variable list or null.
5. statement list must have statement-list statement or null.
  6. statement can be compound statement, assign statement, selection statement, iteration statement, call statement, return statement or input/output statement.
    - a. Selection statement must have if statement or else if statement.
  7. iteration statement must have while statement.
  8. return statement must have return or return expression.
  9. call statement must have call exp.
  10. assign statement must have loc = expression.
  11. expression = expression + expression or expression – expression or expression \* expression or expression / expression or expression && expression or expression || expression or ! expression or - expression | loc = expression or expression < expression or expression <= expression or expression > expression or expression >= expression or expression == expression or expression != expression or (expression)



## **6. Steps that we have followed**

**In the above report we have followed the following steps to make our task easier.**

Step 1: At first we gave a brief description about computers.

Step 2: Then we have defined what programming language is and what its purpose?

Step 3: Then there was an example of programming language which is in C++.

Step 4: In this step we have talked about algorithm. It determines how a program works with minimum time period without any error.

Step5: After that we have described about compiling. It shows how a machine can run a program onto it without any error.

Step 6: Then we have talked about variables. What is variable and types of variables and their usage.

Step 7: It includes a brief description about storage area of a system.

Step 8: Here we have discussed about identifier. Identifier helps us to determine various types of function and variables.

Step 8: In this step we have talked about describes how we can declare something with initialization.

Step 9: Here we have talked about data types. Such as int, bool, float character etc. It will particularly define the data types for a statement.

Step 10: It consists of brief descriptions on CIN and COUT. CIN and COUT is used for input and output.

Step 11: In this section we have talked about input output statement.

Step 12: Here we have described about operators and their types.

Step 13: In this section we have discussed about looping structure such that for loop, if then else, while etc. We have talked about how this loop works, where it is nested, after looping what value it should print.

Step 14: At last we define grammar for a program and determine the mandatory and optional part for the grammar.

## **7. Future Work**

In our thesis we have worked on an electronic learning system. It describes how we can use technology to teach programming to the user. We have made a prototype of our software. In future it can be extended to make a database where all of the lecture plans, user information, and user evaluation result will be stored. There is also possibility of making intelligent software which can generate questions for evaluations artificially. This technique is known as artificial intelligence. It will generate questions automatically based on user performance.

## **8. Conclusion**

This is the implementation idea of the e-learning based programming software tools. This will be web based software and basically develop for the beginner level user. Web Service enabled tools like ASAP used from within virtual learning environment to extend the system usability. This software will ensure that the beginner level user will understand clearly. Hopefully after the pedagogical evaluations that are currently taking place teachers from other institutions will start to use these tools and user communities will develop which will secure the longer term future of these tools. The aim of this software is to make prototype tools that will offer e-learning functionality to the teachers and learners.

## 9. References

- [1]. Cooper, T. (1999). Whose academy is it? *New Statesman*, 128 (4460)
- [2]. Daniel, J. (1996). *Mega universities and knowledge media*. London: Kogan
- [3]. Davies, D. (1998). The virtual university: A learning university. *The Journal of Workplace Learning*, 10 (4), 175 – 213
- [4]. Goddard, A. (1998, November 13). Facing up to market forces. *Times Higher Education Supplement*.
- Goddard, A. (2000, June 16). Big brands key to e-university. *Times Higher Education Supplement*.
- [5]. Hawkes, M. & Cambre, M. (2000). The cost factor. *Technological Horizons in Education*, 28 (1), 26.
- [6]. Katz, R. (2001, May 18). Campus champs tackle heavies. *Times Higher Education Supplement*
- [7]. Kershaw, A. (1996, September/October). People, planning, and process: The acceptance of technological innovation in post-secondary organizations. *Educational Technology*, 44-48.
- [8]. Knight, P. (Ed). (1996). *Assessment for learning in higher education*. London: Kogan Page, SEDA Series.
- [9]. McFadzean, E. (2001). Supporting virtual learning groups. Part 1: A pedagogical perspective. *Team Performance Management*, 7 (3,4), 53-62.
- [10]. National Committee of Enquiry into Higher Education. (2001b). National Report. Chapter 13: Communications and Information Technology. Retrieved November 4th, 2003 from [www.leeds.ac.uk/educol/ncihe/nr\\_202.htm](http://www.leeds.ac.uk/educol/ncihe/nr_202.htm)
- [11]. O'Donoghue, J. & Singh, G. (2001). A study of social-learning networks of students studying an on-line programme. *International Conference on Advanced Learning Technologies (ICALT)*. Madison, Wisconsin USA.
- [12]. Pritchard, A. L. & Jones, D. R. (1996). Global learning. *Open Learning Australia*. Retrieved April 13, 2004 from <http://www.ola.edu.au/paper1.htm>
- [13]. Redfern, S. & Naughton, N. (2002). Collaborative virtual environments to support communication and community in Internet-based distance education.

*Journal of Information Technology Education*, 1 (3), 201-211. Retrieved from <http://jite.org/documents/Vol1/v1n3p201-211.pdf>

[14]. Singh, G. & Priola, V. (2001). Long distance learning and social networks: An investigation into the social learning environment on online students. *Proceedings of the Sixth Annual ELSIN Conference*. 158-164.

[15]. Volery, T. & Lord, D. (2000). Critical success factors in online education. *The International Journal of Education Management*, 14(5), 216 – 223, MCB UP Ltd.