

# Introduction to Programming I

## COS1511

### School of Computing

Revision Notes



Define tomorrow.

© UNISA 2018

UNISA

  
college of  
science, engineering  
and technology

# Introduction

Some key **basic principles** to remember:

- Apply the **BODMAS** rules of Mathematics for all calculations;
- The use of **OR (||)** as well as **AND (&&)** are mathematical concepts from discrete mathematics ( $\vee$ ;  $\wedge$  respectively ), NOT the general English language use.
- Keep together what belongs together.

# Basic Principles

Apply the **BODMAS** rules of Mathematics for all calculations;

- e.g. When asked to calculate the average of three assignment marks.

To do this in a single calculation

- Add the item values (marks) together **in Brackets**;
- **Then divide** by the **number of items** (assignments).

```
float Average = (mark1+mark2 + mark3)/numAssignments
```

# Basic Principles

- Keep together what belongs together.

```
int a = 10;
```

```
a /= 1+1;
```

```
cout << a;
```

You might guess the answer as 11 because most of us translate the compound assignment operator as follows:

```
a = a / 1 + 1; //this is wrong
```

The statement `a /= 1 + 1;` is equivalent to

```
a = a / (1 + 1);
```

What was on the right hand side stays together. So the output is 5, not 11.

# Loops

- The use of OR as well as AND are mathematical concepts from discrete mathematics, NOT the general English language use.
- Know the difference between the two questions below:

1. A loop **should execute** if `age >= 30` or `finalMark <= 65`

2. A loop **should exit** if `age < 30` and `finalMark > 65`

These questions are essentially different ways of asking the same thing. In (1) it says that the loop should execute if either of the conditions is true. Then the condition to exit the loop is obviously the opposite of it. ie., the loop should exit if both these conditions are false.

- See their implementation in the next slide.

# Loops

Suppose we want to input and validate the age of students that qualify for an internship, as well as the final mark obtained for the examination, in a `while` loop. To qualify, the student should be younger than 30 with a final mark of more than 65. The variable names are `age` and `finalMark` respectively. The `while` loop should exit only when the right candidate is found. In other words, if a student does not qualify, the loop should iterate again and continue to validate the age and mark for the next student. Complete the program below.

```
cout << "Enter age: ";  
cin >> age;  
cout << "Enter final mark for exam: ";  
cin >> finalMark;  
while (.....) //complete the condition  
{  
    //complete the while loop  
}
```

# Loops

## **Solution:**

```
int age, finalMark;

cout << "Enter age: ";
cin >> age;

cout << "Enter final mark for exam: ";
cin >> finalMark;

while (age >= 30 || finalMark <=65)
{
    cout << "Sorry, candidate not selected." << endl;
    cout << "Enter age: ";
    cin >> age;
    cout << "Enter final mark for exam: ";
    cin >> finalMark;
}

cout << "Candidate selected." << endl;
```

# Sample loop question

- Suppose two char variables c1 and c2 are initialised and then input repeatedly in the body of a while loop. The loop **has to be executed** as long as the value of c1 is not equal to 'A' or the value of c2 is equal to 'A'. Which of the options below is a correct condition for the while loop?
  1. ((c1 == 'A') || (c2 != 'A'))
  2. ((c1 != 'A') || (c2 == 'A'))
  3. ((c1 == 'A') && (c2 != 'A'))
  4. ((c1 != 'A') && (c2 == 'A'))
  5. None of the above options is correct.

*Correct option: 2*

- The loop has to be executed as long as the value of c1 is not equal to 'A' or the value of c2 is equal to 'A'.
- Therefore, as soon as one of the two conditions is not valid anymore, the loop must stop.
- Look at the wording in the question. The question dictates the conditions for which the loop has to be executed which means the wording can be directly translated to condition.
- However, if the wording is like – the loop has to exit when the value of c1 is equal to 'A' and c2 not equal to A, then it needs a bit more effort to arrive at the condition. See the next question.



# Sample loop question contd...

- Suppose two char variables c1 and c2 are initialised and then input repeatedly in the body of a while loop. The loop **has to exit** when the value of c1 is equal to 'B' and the value of c2 is not equal to 'B'. Which of the options below is a correct condition for the while loop?
  1. ((c1 == 'B') && (c2 != 'B'))
  2. ((c1 == 'B') || (c2 != 'B'))
  3. ((c1 != 'B') && (c2 == 'B'))
  4. ((c1 != 'B') || (c2 == 'B'))
  5. None of the above options is correct.

*Correct option: 4*

- Contrary to the previous question, where the conditions under which the loop must be executed, is described, the conditions under which the loop must exit, are described in this question. The loop must exit when c1 is equal to 'B' and c2 is not equal to 'B'. **Therefore, the condition in the loop should be 'opposite' of this.**
  - Opposite of (c1 is equal to 'B') → c1 != 'B'
  - Opposite of (c2 not equal to B) → c2 == 'B'
  - Opposite of 'AND' which connects the above two conditions → OR
- Combine all, and you have the answer!

# More Topics

1. Variable diagrams;
2. Functions;
3. Data structures (Arrays & Structs); and
4. Two-dimensional arrays;
5. Reference parameters.

Thank you to all the students who participated.

# 1. Variable Diagrams

Variable diagrams help in tracing the value stored in each variable as the program executes.

In a variable diagram we only show those lines which changes the value of variable(s).

For instance, we do not draw diagrams for `cout` statements.

# 1. Variable Diagrams- Example

Consider the following C++ code segment below

```
1 int result(int valueP)
2 {
3     int a = 2;
4     int count = 0,
5     while (count < valueP)
6     {
7         a += count + a / 2,
8         count += 2,
9     }
10    return a,
11 }
```

The variable diagram should indicate what changes took place in variables.

We DO NOT represent every single line of a program in a variable diagram; it is only those lines where a variable is declared and/or changes made to them represented.

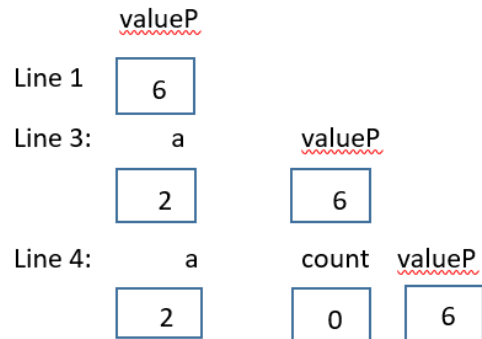
The most simplified way to work out the variable diagrams is to label the line numbers and substitute the values of the variables as you go along; e.g.:

```
1.  valueP = 6;
2.  {
3.  a = 2;
4.  count = 0;
5.  while ( 0 < 6) // count = 0; valueP = 6; count = 2; count = 4;
   //count = 6 then loop exits;
6.  {
7.  a = a + count + a / 2 ;
   // a = 2 + 0 + (2 / 2) = 3;
   a = 3 + 2 + (3 / 2) = 3 ; a = 3 + 4 + 3 / 2 ; a = 0 + 2 / 2 ;
8.  count = 0 + 2; count = 2; count = 4; count = 6;
9.  }
10. return a; //current value of a after line 7 updated
11. }
```

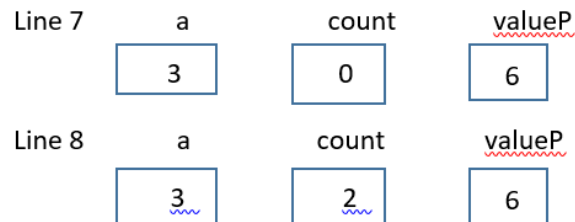
Demonstrate the execution and output of the program by drawing a variable diagram that traces each line of code if the value of valueP is 6. You are required to draw a variable diagram to illustrate what the code does

**Note: The quality of the past exam papers uploaded on myunisa is very poor. Some of the semicolons are displayed as commas. Consider them as semicolons, especially those at the end of C++ statements.**

# Solution to previous example



We do not have to represent line 5 and line 6. Line 5 just checks a condition and line 6 is a curly bracket.



After line 9, control goes back to line 5, to check the condition again.

Lines 7 and 8 repeat in the variable diagram until the while loop exits.



## Solution contd....

Line 8	a	count	<u>valueP</u>
	6	4	6
Line 7	a	count	<u>valueP</u>
	13	4	6
Line 8	a	count	<u>valueP</u>
	13	6	6

On checking the condition, it is false now, so the function exits.

The value of a that is returned is 13.

# 1. Variable Diagrams- Example

Consider the C++ code segment below What value will `newval` have after this code has been executed?  
(2)

```
int var1 = 4,  
int var2 = 10,  
int newval = 0;  
if (var1 * 2 >= var2)  
    newval = 5 + 2 * var2;  
else if (var1 < var2)  
    newval = var2 - var1 * 2,  
else  
    newval = var1;
```

**Solution:**

	var1		
Line 1	4		
Line 2:	var1	var2	
	4	10	
Line 3:	var1	var2	<u>newval</u>
	4	10	0
Line 7:	var1	var2	<u>newval</u>
	4	10	2

- You can use variable diagrams to trace the value as it changes.
- The variable diagram should indicate those lines where changes took place to variable(s).
- In this question keep in mind that only statements following the condition that is true will execute, not all of them.
- `if(var1 * 2 >= var2)` evaluates to `if(4 * 2 >= 10)`. It is false, therefore the control goes to the next `else if`. There it checks the condition `if(var1 < var2)` which evaluates to `if(4 < 10)`. The condition is true, therefore the line of statement following this `else if` will execute and the control will exit the `if` structure.
- Therefore after line 3 in the VD, we need to represent only line 7.

## 2. Functions

- A function is a piece of code(a group of statements) that executes a particular task.
- All C++ programs have at least one function in it which is the main().
- Two types of functions:
  - built-in functions. eg: abs(), rand(), sqrt()
  - user-defined functions



## Functions contd...

- Built-in functions:

C++ has built-in functions for which we do not have to write the code. eg: rand(), abs(), sqrt() etc. These functions are already defined in the C++ library. We just have to invoke them through a function call.

- User-defined functions

Users can define their own function. It takes the form:

```
return-type  function-name(parameter list)
{
body of the function;
}
```

### 3. Data Structures- Arrays & Structs

- A **data structure** is a collection of related data values stored under a single name and organised so that the individual values can be accessed separately.
- **Structs** are data structures for storing related data values of different types.
- **Arrays** are **data structures** for storing related data values of the same type.

## 4. Data Structures-2D Arrays

- Two-dimensional Arrays
- eg: `int arr[2][3];` declares a 2D array `arr` with two rows and 3 columns.
- When a 2D array is initialized as follows the first dimension is optional.
- eg:- `int arr[][3] = { {1,2,3}, {4,5,6} };`

# Solution to Q 6.2.1 of Oct/Nov 2016 exam paper

```
bool checkNumber(const int arr1[size_of_array], int num1, int & pos1)
{
    bool flag = false;
    //checking the number in the array
    for(int i=0;i< size_of_array;i++)
    {
        if(arr1[i]==num1)
        {
            pos1=i;
            flag=true;
        }
    }
    return flag;
}
```

## 5. Reference Parameters

- Reference parameters are used when the values for variables declared in the `main` function should be updated within other functions.
- For example, if a function has to change more than one value and if both the values has to be returned to the calling function – then we have to use reference parameters (&).

# 5. Reference Parameters - example

```
#include <iostream>
using namespace std;

void twice (int &firstP, int &secondP)
{
    firstP = firstP * 2;
    secondP = secondP * 2;
}

int main()
{
    int first, second;
    cout << "Enter the first number to double: " << endl;
    cin >> first;
    cout << "Enter the second number to double : " << endl;
    cin >> second;

    twice(first, second);
    cout << "The first number times two = " << first << endl;
    cout << "The second number times two = " << second << endl;

    return 0;
}
```

Reference parameter variables declared with the ampersand (&).

Here, the value of the variable `first` in the main function is updated. The same happens for the variable `second`, in the next line.

The function `twice` is called with the two variables that it should update.

The updated variable values after the execution of the function `twice` will be displayed.

# 5. Reference Parameter example Code

```
#include <iostream>
using namespace std;

void twice (int &firstP, int &secondP)
{
    firstP = firstP *2;
    secondP = secondP * 2;
}

int main()
{
    int first, second;
    cout<< "Enter the first number to double: "<<endl;
    cin >> first;
    cout<< "Enter the second number to double : " << endl;
    cin >> second;

    twice(first, second);
    cout << "The first number times two = " << first << endl;
    cout << "The second number times two = " << second << endl;

    return 0;
}
```

# Sample question

## Question 2

The output of the following program will be:

```
#include <iostream>
using namespace std;
int main()
{
    cout << "Hello there, everybody!";
    cout << "I'm writing COS1511." << endl;
    cout << "Goodbye." << endl;
    return 0;
}
```

- 1- *Hello there, everybody! I'm writing COS1511.Goodbye.*
- 2- *Hello there, everybody!*  
*I'm writing COS1511.*  
*Goodbye.*
- 3- *Hello there, everybody! I'm writing COS1511.*  
*Goodbye.*
- 4- None of the above

Answer is option 3.

There is no `endl` in the first `cout` statement. Hence the string in the second `cout` statement will be printed on the same line as the previous string.

There is `endl` in the second `cout` statement. So the third string will be printed on a newline.



# Sample Question

```
int x = 7,  
if (x > 10)  
    cout << "High";  
    cout << "Low",
```

- [In your past exam papers because of the poor quality, you are seeing commas where it should be semicolon in some places].

The indentation is given just to confuse you.

Here `x` is not greater than 10. so the statement after the `if` structure will be executed.

- Output is `Low`



# Thank you

Define tomorrow.

UNISA

