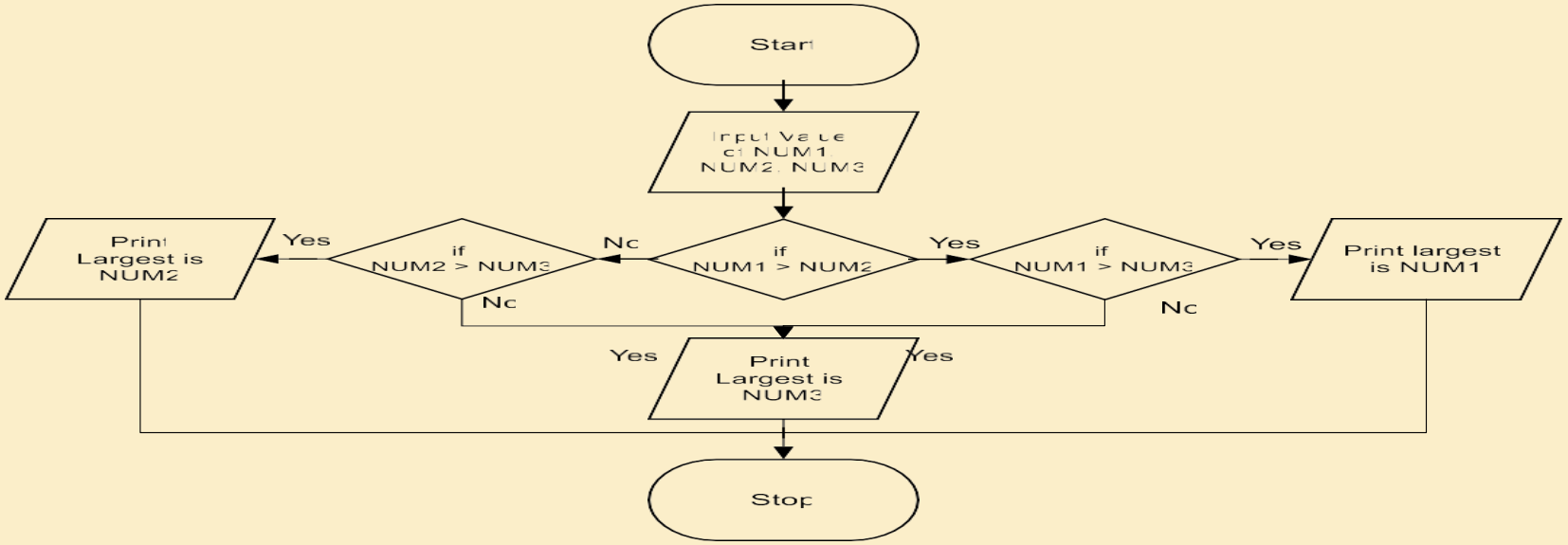
Introduction to C++
COMP101



Dr. Zaid Ameen

Algorithm & Flowchart

COMP101

A definition of a computer is an electronic device that can input data, process data and output data, accurately and at great speed. Data are any kind of information that can be codified in some manner and input into the computer.

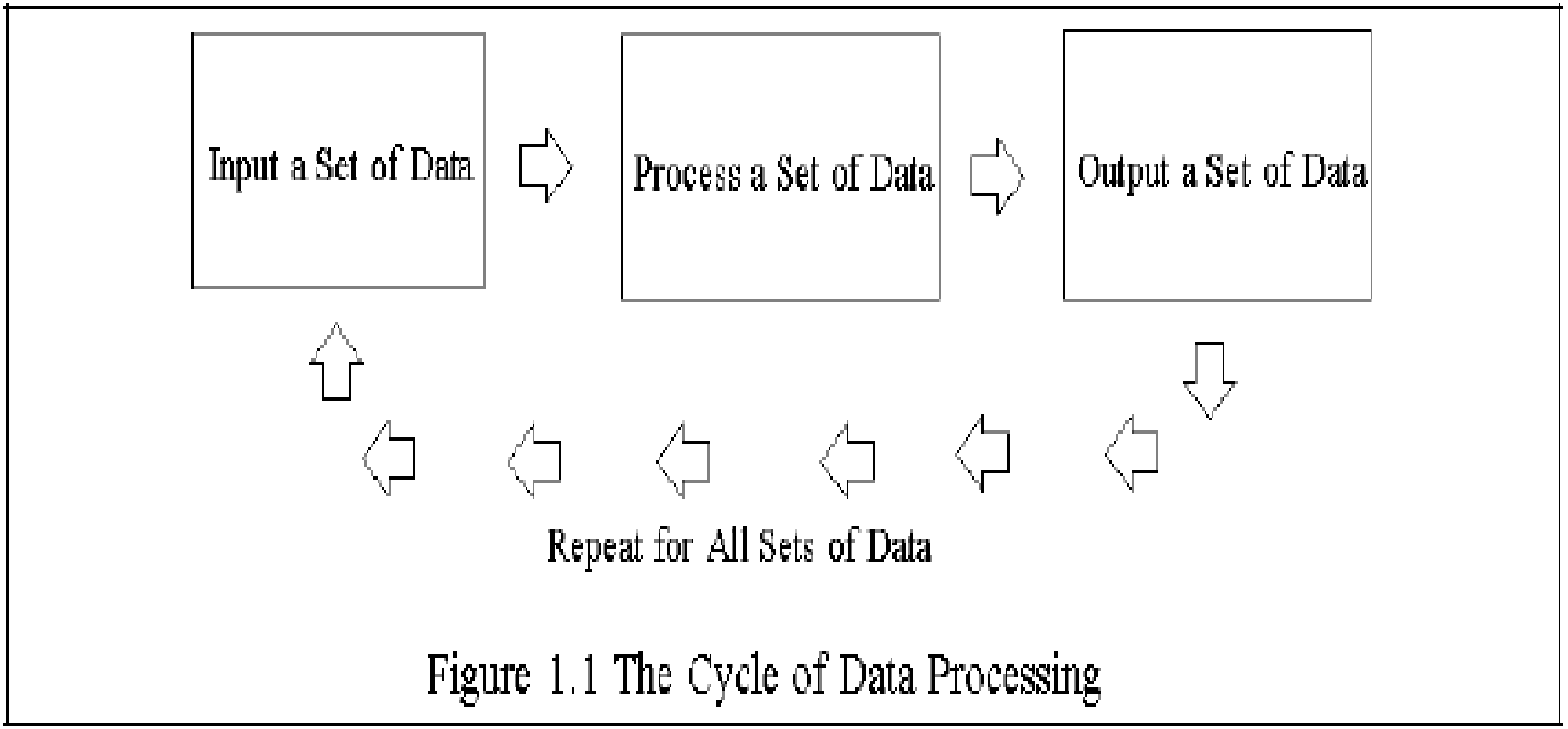


Figure 1.1 The Cycle of Data Processing

PROGRAM

What is a program? A computer program is a series of instructions that tell the computer every step to take in the proper sequence in order to solve a problem for a user. For example, "*If You want to compute your average at last year in high school, you must input your scores for each class, sum all scores and then compute average by dividing summation on the number of classes. Finally, print the result on screen*". Thus, I have a new term for programs that have one or more errors in them — "mostly working software." When a program has an error in it, that error is often called a "**bug**." And the process of getting all of the errors out of a program is called **debugging**.

Nearly every significant program follows the same fundamental design and it is called the **Cycle of Data Processing**

The computer internally operates on the binary number system. In the binary number system, the only valid digits are 0 and 1. For example, if you add in binary 1 plus 1, you get 10, just as if you added 9 + 1 => 10 in the decimal or base 10 system. Why does the computer use binary? Electronic circuits can either have some electricity in them or not. If a circuit element, such as a transistor, has electricity, it can be said to contain a 1; if none, then a 0. This is the basis for computer operations. The actual instructions that make up a program are all in binary, long strings of binary digits. But no one wants to try to write out these long tedious series of 1's and 0's to try to direct the computer to solve a problem. Rather a high-level language is used. In this case, we use the C++ language. In a high-level language, we use various symbols and mathematical notations to create the program which is called the **source program**. A *source program* is the precise series of high- level language statements in the proper order for the computer to follow to solve the problem. For us, that source file has the file extension of **.cpp**.

Another piece of software called the **compiler** inputs our source program and converts it into the machine language, binary equivalent. If you make some typos, these show up as syntax errors when the compiler tries to convert the source program. A *syntax error* just means that you have coded the C++ instruction incorrectly. When you first compile a program and suddenly see a large number of compile errors, don’t panic. Often it is just one small syntax error that cascades into many errors. Fix the original error and the others are automatically fixed. The output from a successful . The **obj** file contains the binary machine language instructions to control the computer in solving your problem. However, it is not the final program; object files are missing something.

1.1 Variables and Constants

A variable in C++ is a modifiable data object, that is, it is a place to store data and the value stored there can change as we desire. For example, In C++, a data object is a region in memory in which to store data. A data value is the contents of that object. A place to hold the quantity purchased, say 42, could be called qty but the actual contents of qty, its data value, would be 42 in this case.

The opposite of a variable is a constant data object. With a **constant** data object, once the initial value is defined, *it can never be changed in any way*. Additionally, if we want to defined a constant to hold the value of **PI**, it should be a constant as well, 3.14159.

When programming, we store the variables in our computer's memory, but the computer has to know what kind of data we want to store in them, since it is not going to occupy the same amount of memory to store a simple number than to store a single letter or a large number, and they are not going to be interpreted the same way. The memory in our computers is organized in bytes. A byte is the minimum amount of memory that we can manage in C++. A byte can store a relatively small amount of data: one single character or a small integer (**generally an integer between 0 and 255**). The syntax to define a variable is actually quite simple.

datatype variable name;

In addition, the computer can manipulate more complex data types that come from grouping several bytes, such as long numbers or non-integer numbers. We display a summary of the basic fundamental data types in C++, as well as the range of values that can be represented with each one:

Name	Description	Size*	Range*
char	Character or small integer.	1byte	signed: -128 to 127 unsigned: 0 to 255
short int (short)	Short Integer.	2bytes	signed: -32768 to 32767 unsigned: 0 to 65535
int	Integer.	4bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
long int (long)	Long integer.	4bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
bool	Boolean value. It can take one of two values: true or false.	1byte	true or false
float	Floating point number.	4bytes	+/- 3.4e +/- 38 (~7 digits)
double	Double precision floating point number.	8bytes	+/- 1.7e +/- 308 (~15 digits)
long double	Long double precision floating point number.	8bytes	+/- 1.7e +/- 308 (~15 digits)
wchar_t	Wide character.	2 or 4 bytes	1 wide character

(that is 3.4 x 10 power 38)

34000 = 3.4E4 (or 3,4E4)
5000 = 5E3
0.005 = 5E-3

Declaration of variables

In order to use a variable in C++, we must first declare it specifying which data type we want it to be. The syntax to declare a new variable is to write the specified of the desired data type (like int, bool, float...) followed by a valid variable identifier. For example:

int a; float mynumber;

These are two valid declarations of variables. The first one declares a variable of type **int** with the identifier a. The second one declares a variable of type float with the identifier **mynumber**. Once declared, the variables **a** and **mynumber** can be used within the rest of their scope in the program. If you are going to declare more than one variable of the same type, you can declare all of them in a single statement by separating their identifiers with commas. For example: **int a,b,c; "The three variables (a, b and c) are int type, and has exactly the same meaning as:**

int a; int b; int c;

The integer data types *char*, *short*, *long* and *int* can be either signed or unsigned depending on the range of numbers needed to be represented. Signed types can represent both positive and negative values, whereas unsigned types can only represent positive values (and zero). This can be specified by using either the specifier *signed* or the specifier *unsigned* before the type name. For example:

unsigned short int NumberOfSisters;

signed int MyAccountBalance;

By default, if we do not specify either *signed* or *unsigned* most compiler settings will assume the type to be signed, therefore instead of the second declaration above we could have written:

int MyAccountBalance;

Finally, signed and unsigned may also be used as standalone type specifiers, meaning the same as

signed int and unsigned **int** respectively. The following two declarations are equivalent:

unsigned NextYear;
unsigned int NextYear;

What are the rules for names in C+ + ?

1. Names can be from one character long to as many characters as desired; however, only the first thirty-one characters are used by the compiler.

2. The name must begin with a letter of the alphabet or the _character. (However, names beginning with an _generally have a special purpose meaning and should be avoided.)

3. Each name must be unique.

4. Remember also that C+ + is case sensitive.

5. The _character can be used to separate compound names.

6. A blank cannot be used in a variable name because a blank is a form of white space and white space is the delimiter between language elements.

7. Numerical digits can be a part of the name after the first character of the name.

8. There are some reserved words in C+ + and these words are the language “verbs” and components. A

variable cannot be a reserved word.

9. All variable names in a program must be meaningful names.

Where are the variable definitions placed within a program? The answer is that a variable must be defined before its first use in an instruction. Normally, the variables to be used for input, those for calculation results and those for output are defined at the beginning of the main() function. For example, suppose that our program needed to define variables for a cost, a quantity and a total. The program up to this point is as follows.

```
#include <iostream>

#include <iomanip>


using namespace std;

int main () {

int qty; // quantity ordered
double cost; // cost of one item
double total; // total cost of the order
```