RECORDS:

A record is a user defined data type suitable for grouping data elements together. All elements of an array must contain the same data type.

A record overcomes this by allowing us to combine different data types together. Suppose we want to create a data record which holds a student name and mark. The student name is a packed array of characters, and the mark is an integer.

We could use two seperate arrays for this, but a record is easier. The method to do this is,

• define or declare what the new data group (record) looks like

• create a working variable to be of that type.

Defining a Record

To define a record type, you may use the type declaration statement. The record type is defined as:

```
type
record-name = record
field-1: field-type1;
field-2: field-type2;
...
field-n: field-typen;
end;
```

Here is the way you would declare the Book record:

```
type
Books = record
title: packed array [1..50] of char;
author: packed array [1..50] of char;
subject: packed array [1..100] of char;
```

```
book_id: integer;
end;
```

The record variables are defined in the usual way as:

```
var
r1, r2, ... : record-name;
```

Alternatively, you can directly define a record type variable as:

```
var
Books : record
title: packed array [1..50] of char;
author: packed array [1..50] of char;
subject: packed array [1..100] of char;
book_id: integer;
end;
```

The following portion of code shows how to define a record, then create a working variable to be of the same type.

```
TYPE
studentname = packed array[1..20] of char;
studentinfo = RECORD
name : studentname;
mark : integer
END;
VAR student1 : studentinfo;
```

The first portion defines the composition of the record identified as *studentinfo*. It consists of two parts (called **fields**).

The first part of the record is a packed character array identified as *name*. The second part of *studentinfo* consists of an integer, identified as *mark*. The declaration of a record begins with the keyword **record**, and ends with the keyword **end;**

The next line declares a working variable called *student1* to be of the same type (ie composition) as *studentinfo*.

Each of the individual fields of a record are accessed by using the format,

recordname.fieldname := value or variable;

An example follows,

```
student1.name := 'JOE BLOGGS '; {20 characters}
student1.mark := 57;
```

Lets create a new data record suitable for storing the date

```
type date = RECORD
day : integer;
month : integer;
year : integer
END;
```

This declares a **NEW data type** called *date*. This *date* record consists of three basic data elements, all
integers. Now declare working variables to use in the program. These variables will have the same
composition as the *date* record.

```
var todays_date : date;
```

defines a variable called *todays_date* to be of the same data type as that of the newly defined record *date*.

ASSIGNING VALUES TO RECORD ELEMENTS

These statements assign values to the individual elements of the record *todays_date*,

```
todays_date.day := 21;
todays_date.month := 07;
todays_date.year := 1985;
```

SELF TEST

What does this statement do?

readln(todays_date.day, todays_date.month, todays_date.year);

Answer:

SELF TEST

What does this statement do?

readln(todays_date.day, todays_date.month, todays_date.year);

Self Test ..

The program statement reads three values from the keyboard, into each of the individual fields of the record todays_date.