الباب الرابع

معالجة البيانات

Data Processing

4–1 مقدمة

تستخدم العبارة (معالجة البيانات Data Processing) للتعبير عن عملية تحويل البيانات من شكل إلى آخر أكثر وضوحاً أو فائدة . فمثلاً قد تكون لدينا مجموعة من البينات عن أسماء الموظفين ومرتباتهم الأساسية وعلاواتهم وخصمياتهم، ولكن هذه البيانات تبقى بدون قيمة عملية حتى نعالجها ، وكمثال على المعالجة عملية حساب صافي مرتب كل موظف .

ومن أهم وظائف الحاسوب معالجته للبيانات ذات الحجم الكبير في وقت قصير . أضف إلى ذلك تجنب الوقوع في الأخطاء الحسابية ، وغيرها من الأخطاء البشرية ، ولذلك يوصف الحاسوب بأنه آلة لمعالجة البيانات .

في هذا الباب ، نتعرض لخصائص لغة باسكال في معالجة البيانات، ويشمل ذلك أنواع البيانات وتركيبها وتخزينها في الذاكرة الإضافية (الأقراص) والذاكرة الرئيسية .

4–2 النوع المسرود Enumeration Type

نلاحظ أن الأنواع الأربعة للبيانات التي درسناها حتى الآن لها نطاق معين لا تخرج منه :

فالنوع INTEGER نطاقه الأعداد الصحيحة (الموجبة والسالبة) ، والنوع BOOLEAN نطاقه قيمتان فقط ، هما (TRUE و FALSE)، والنوع CHAR يشمل نطاقه كل الرموز الأبجدية الصغيرة والكبيرة والأعداد

والإشارات ، أما النوع REAL فيشمل نطاقه جميع الأعداد الحقيقية بما في ذلك الكسور العشرية .

ولكن في كثير من الأحيان نحتاج إلى أنواع أخرى من النطاق ؛ فإذا كان المتغير يعبر مثلا عن أسماء أيام الأسبوع ، فقد يكون من الأفضل تحديد نطاقه مثلاً بالفئة :

WEEK = (Sat, Sun, Mon, Tue, Wed, Thu, Fri)

أو كمثال آخر قد يعبر متغير ما عن فئة الألوان التالية:

COLORS = (White, Black, Red, Blue, Green, Yellow)

وهكذا نستطيع أن نحصر أمثلة متعددة لهذا النوع من المتغيرات ، وهذا ما جعل مصمم لغة باسكال يمكننا من تحديد أي نوع نريده من المتغيرات .

فإذا كان المتغير Day يعبر عن اسم لأحد أيام الأسبوع ، نستطيع تحديد ذلك في بداية البرنامج بالجملة التالية :

Day: (Sat, Sun, Mon, Tue, Wed, Thu, Fri)

أو بطريقة أخرى نعرف أولاً النوع :

TYPE WEEK = (Sat, Sun, Mon, Tue, Wed, Thu, Fri);

ثم نحدد أن المتغير Day ينتمي إلى هذا النوع

VAR Day: WEEK;

حيث d عدد صحيح ، بدلاً من المتغير من النوع Week ، و لكن الجملة :

FOR day: = Sat TO Fri DO

ذات معنى أكثر وضوحاً .

شكل (1–2–4) : برنامج يستخدم النوع المسرود .

```
PROGRAM weeklyTotal;
TYPE week=(Sat, Sun, Mon, Tue,Wed,Thu,Fri);
VAR day: week;
    de, total : INTEGER;
BEGIN
     total:+0;
     FOR day:=Sat TO Fri DO
     BEGIN
          WRITE('enter amount-->');
          READ(de);
          total := total + de;
     END;
     WRITELN('TOTAL=',total);
END.
```

كما تجب الملاحظة هنا بعدم جواز قراءة أو كتابة متغير من النوع المسرود ، مثل :

```
WRITELN ( Day );
```

وعلى المبرمج تفادي مثل هذه الجملة إذا أراد طباعة اسم اليوم ، كأن يكتب مثلا :

```
IF ( Day = Sat ) THEN
          WRITELN ( ' Sat ' );
```

النطاق الجزئي : Subrange

قد يحدث أحياناً أن يكون لدينا متغير ذو نطاق جزئي من نوع تم تحديده .

فمثلاً إذا كان المتغير x من النوع الصحيح INTEGER بحيث 0 ? X ? 100

فإن الجملة :

VAR x: 0.. 100;

تجعل لهذا المتغير نطاقاً جزئيًّا من النطاق الكلي INTEGER . بحيث إذا خرجت قيمة x عن هذا النطاق (أي من الصفر إلى 100) يتوقف التنفيذ معلناً رسالة خطأ (out of range) أي الخروج عن النطاق. وكمثال آخر، نفترض أن درجة الطالب في مادة ما تعطى بالحروف مثل : A (ممتاز) ، B (جيد جداً) ، و C (جيد) و D (مقبول) و E (ضعيف) . يمكننا تحديد النطاق في هذه الحالة على النحو التالي :

VAR Grade: 'A' …'E';

بهذه الطريقة يمكننا اكتشاف الخطأ في بيانات الدرجة إذا حدث وأن خرجت عن إحدى الحروف من A إلى E .

كما يجوز أن نحدد جزءاً من نطاق نوع مسرود . فمثلاً إذا كان MONTHS يمثل أشهر السنة الميلادية :

TYPE MONTHS = (Jan, Fab, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec);

وكان Spring متغيراً يمثل أشهر الربيع (Mar, Apr, May) ، فيمكننا تحديد ذلك بواسطة الجملة :

VAR Spring: Mar … May;

ونلاحظ هنا إمكانية استخدام المتغير spring في جملة FOR كالمثال الآتي:

```
FOR Spring: = Mar TO May DO
BEGIN
  IF Spring = Mar THEN WRITELN ( ' March' );
  IF Spring = Apr THEN WRITELN ( ' April ' );
  IF Spring  =  May THEN WRITELN ( 'May' );
END;
```

وينتج عن هذه الجملة طباعة الآتي :

March

April

May

وتجدر الإشارة هنا إلى وجود دوال خاصة بالنوع المسرود نحتاج إليها في كثير من الأحيان ، وهي :

1- الدالة Succ (اختصار لكلمة Successor أي الموالي) وهي تعطـــى العنصـــر الموالي في الترتيب . فمثلاً نسبة إلى النوع المسرود week نجد أن :

Succ (Sat) هو Sun

Succ (Sun) هو Mon

وهكذا ...

أما Succ (Fri) فهو غير محدد لأن Fri آخر عنصر في النوع :

Week = (Sat, Sun, Mon, Tue, Wed, Thu, Fri)

2- الدالة Pred (اختصار لكلمة Predecessor أي السابق) ، وهي تعطي العنصر السابق في الترتيب فمثلاً في النوع week :

Pred (Sun) هو Sat

Pred (Mon) هو Sun

ولكن .

Pred (Sat) غير محدد لأن Sat هو أول عنصر في النوع Week .

3- الدالة Ord (وهي اختصار لكلمة ordinal number أي رقـــم الترتيب) وهي تعطي موقع العنصر في النوع من حيث الترتيب. فمثلا في النوع week المذكور أعلاه نجد أن :

Ord(Sat) هو صفر

Ord(Sun) هو 1

و هكذا ...

لاحظ إمكانية استخدام الدوال الثلاثة المذكورة على النـــوع CHAR أو INTEGER . فمثلاً ('A') Succ هو 'B' ، و (6) Pred هو 5 .

3-4 المصفوفات ذات البعد الواحد

تستخدم المصفوفة لتخزين مجموعة من البيانات مـــن النـــوع نفســـه في الذاكرة الرئيسية بطريقة تمكننا من استرجاعها و الوصول إليها مباشرة . فقـــــد نحتاج مثلاً لتخزين أسماء أيام الأسبوع حسب التسلسل :

```
Day [1]: = ' Sat ' ;
Day [2]: = ' Sun';
...
Day[7] := 'Fri' ;
```

وبذلك يصبح المتغير Day ذا سبع قيم مخزنة في الذاكرة ، و نشير إلى كل واحد منها بترتيبه بين هذه القيم ، وهذا ما يعرف بتركيبة المصفوفة . فمثـــلا إذا كان:

```
x[1] := 5 ;
x[2] := 9;
x[3] := 33;
```

فإن لدينا هنا مصفوفة اسمها x وتتكون من 3 عناصر ، العنصر الأول هو 5، والعنصر الثاني هو 9، والعنصر الثالث هو 33 . ويمكننا هنا التعبير عن عناصر

المصفوفة باستخدام دليل من النوع الصحيح ، وليكن مثلاً k ، أي إن x[k] هـو العنصر رقم k من المصفوفة x .

أي أن المصفوفة تنسيق لعدد من البيانات ذات النوع الواحد تخـزن في الذاكرة الرئيسية . ويتم تخزين هذه البيانات تحت اسم واحد هو اسم المصفوفة، وللتمييز بين عنصر وآخر يكون للمصفوفة دليل (من النوع الصحيح أو المسرود) يبين موقع العنصر في المصفوفة .

بهذه الوسيلة يمكن الوصول مباشرة إلى أي عنصر من عناصر المصفوفـة دون المرور على ما قبله من عناصر ، و لهذا توصف المصفوفة بأنها ذات تركيبـة وصول مباشرة direct - access structure .

نبدأ أولاً بمفهوم المصفوفة ذات البعد الواحد (أي ذات العمود الواحد) One - dimensional array ، وهي المصفوفة التي تملك دليلا واحدا .

يحدد نوع المتغير بأنه مصفوفة ذات بعد واحد في بداية البرنامج . فقـد يكون لدينا على سبيل المثال الجملة :

```
VAR  price: ARRAY [1..5] OF REAL;
```

هذه الجملة تبين أن price هو اسم لمصفوفة ذات بعد واحد ، متكونـة من 5 عناصر ، هي :

price [1], price [2], price [3], price [4], price [5]

وهذه العناصر أعداد كسرية real . عند تنفيذ هذه الجملة نكـون قـد حجزنا في الذاكرة الرئيسية 5 مواقع لهذه العناصر الخمسة . وهذا الحجز طبعاً هو حجز مؤقت ينتهي بانتهاء تنفيذ البرنامج .

وكمثال آخر ، نعتبر الدليل من النوع المسرود (items) حيث :

```
TYPE items = ( chair, table, desk );
VAR price : ARRAY  [items] OF REAL;
```

بهذه الطريقة يتضح أن المصفوفة price لها دليل من النوع items وهو عبارة عن فئة تتكوم من 3 عناصر فقط. بهذا نتمكن في البرنامج من استخدام جمل واضحة المعنى مثل :

```
price [chair]: = 25.5;
price [table]: = 56.0;
price[desk] := 123;
```

كما يمكن أن تكون المصفوفة من إحدى الأنواع التي درسناها للمتغيرات ، أي المنطقية والصحيحة والرمزية ... فمثلاً :

```
VAR number: ARRAY [1..25] OF INTEGER;
```

جملة تفيد بأن number هو اسم لمصفوفة أحادية البعد ، متكونة من 25 عنصر من النوع العددي الصحيح .

مثال (1–3–4) :

تتبع البرنامج المبين بالشكل (1-3-4) وبين الغرض منه .

يوجد بهذا البرنامج مصفوفتان هما x و diff و الحد الأعلى لعدد العناصر لكل منهما هو 50 . المتغير n يعبر عن عدد العناصر المستخدمة من هذه المصفوفات في هذا البرنامج وليكن 8 . تتم قراءة عناصر x الثمانية باستخدام جملة FOR - DO ثم حساب مجموع هذه العناصر وهو sum ، وبالتالي نحسب المتوسط بالقسمة على n .

شكل (1-3-4) : حساب الانحراف عن المتوسط باستخدام المصفوفة .

```
PROGRAM stat;
VAR x, diff : ARRAY[1..50]OF REAL;
    sum, ave : REAL;
    i, n : INTEGER;
BEGIN
     WRITE('Enter number of items-->');
```

```
READ(n);
FOR i:= 1 TO n DO
BEGIN
     WRITE('Enter x',i,'-->');
     READ(x[i]);
END;
sum:=0;
FOR i:=1 TO n DO
    sum := sum + x[i];
ave := sum/n;
WRITELN('average=',ave:5:2);
FOR i:=1 TO n DO
    diff[i]:=x[i]-ave;
WRITELN('i     x       diff');
FOR i:=1 TO n DO
    WRITELN(i,'    ',x[i]:5:2,
 '  ',diff[i]:5:2);
END.
```

أما المصفوفة diff فواضح من البرنامج أنها تحمل الفرق بين كل عنصر والمتوسط . و أخيراً تتم طباعة المصفوفتين x و diff .

إن الغرض من هذا البرنامج هو إيضاح ضرورة استخدام المصفوفات في كثير من الحالات ، وهي بالتحديد الحالات التي تستدعي تخزين البيانات في الذاكرة الرئيسية نظراً لحاجتنا إليها في أماكن متعددة من البرنامج . فبعد أن تم حساب المتوسط عناصر x ، احتجنا إلى هذه العناصر لحساب الفرق بينها وبين المتوسط. وبدون تخزينها في مصفوفة ، نضطر إلى قراءتها من لوحة المفاتيح من جديد .

4-4 ترتيب القوائم Sorting

من أحد التطبيقات المهمة للمصفوفات أنها تمكننا من حل مسألة ترتيب قائمة من الأرقام أو الأسماء . لتوضيح المطلوب ، نفترض أن لدينا قائمة من أربعة أرقام :

52
43
65
23

واضح أن هذه القائمة غير مرتبة تصاعدياً ، وأن الترتيب التصـــــاعدي يكون على النحو :

23
43
52
65

هناك عدة طرق لترتيب القوائم ، ويعتبر هذا الموضوع مـــن المواضيـــع المهمة في تطبيقات الحاسب الآلي . سوف نستخدم هنا إحدى الطرق ، ولعلـــها أبسطها ، وتعتمد هذه الطريقة على الخطوات التالية :

1- أوجد موضع الرقم الذي يجب أن يكون في رأس القائمة .

2- ضع هذا الرقم في رأس القائمة ، وضع مكانه الرقم الذي كان يشـــغل ذلك المكان الأول .

3- كرر هذا العمل بالنسبة للرقم الذي يأتي في الـــترتيب الثـــاني وذلـــك باستبداله مع الرقم في الموقع الثاني .

استمر في هذا العمل حتى نهاية القائمة .

هذه الخطوات غير دقيقة لكتابة البرنامج ، ولكنها تساعدنا في تتبعه وفهمه.

ونقدم هنا ملاحظات حول البرنامج المبـــين بالشـــكل (1-4-4) ، والذي يعتمد على هذه الفكرة في ترتيب درجات الطلبة ترتيباً تصاعدياً :

شكل (1-4-4) : برنامج الترتيب التصاعدي .

```
PROGRAM sort;
VAR gr:ARRAY[1..100]OF REAL;
    first, temp: REAL;
    n, i, k, p : INTEGER;
BEGIN
     WRITE('Enter number of items-->');
     READ(n);
```

```
FOR i:=1 TO n DO
BEGIN
     WRITE('Enter grade',i,'-->');
     READ(gr[i]);
END;
FOR i:=1 TO n-1 DO
BEGIN
     first:=gr[i];
     p:=i;
     FOR k:=i+1 to n DO
          IF gr[k] < first THEN
          BEGIN
             first:= gr[k];
             p:=k;
          END ;
     temp:=gr[i]; gr[i]:=gr[p];
     gr[p]:=temp;
END;
FOR i:=1 TO n DO
    WRITELN(gr[i]:5:2);
END.
```

- عدد العناصر المطلوب ترتيبها هو n 100
- نخزن هذه العناصر في المصفوفة grade
- نسمي العنصر الذي بأخذ الترتيب الأول first
- نسمي موضع العنصر الأصغر في القائمة p .
- نجري عملية وضع العنصر الأصغر في الترتيب الأول في الدورة الأولى، والترتيب الثاني في الدورة الثانية ، وهكذا ، إلى (n -1) من الدورات ، ولانجري الدورة n لأن العنصر المتبقي هو طبعاً الأكبر .

5-4 المصفوفات ذات البعدين

يمكن اعتبار القائمة (العمود) بأنها مصفوفة ذات بعد واحد ، ويتحدد موقع العنصر فيها باستخدام دليل واحد (index) ، أما الجدول (table) فيتكون من عدد من الصفوف والأعمدة ، ولتعين موقع أي عنصر في الجدول،

نحتاج إلى دليلين ، واحد لتحديد الصف والثاني لتحديد العمود ، لذلك نسمي الجدول مصفوفة ذات بعدين .

مثال (1–5–4)

إذا كان الجدول التالي يبين عدد الطلبة في كلية العلوم حسب الأقسام و السنة الدراسية :

السنة	الحاسوب	احصاء	
Year	**Computer**	**Stat**	**Math**
1	**52**	**46**	**75**
2	**43**	**62**	**55**
3	**35**	**17**	**34**
4	**22**	**9**	**40**

ورمزنا لهذا الجدول بالمصفوفة student فإن :

student [year, dept]

تعني عدد الطلبة في السنة (year) بالقسم (dept). فمثلاً :

student [1, computer] = 52
student [2, stat] = 62
student [3, stat] = 17

نلاحظ وجود دليلين للمصفوفة student هما year و dept : الأول عددي صحيح و الثاني مسرود . وبالتالي فإن المصفوفة student هي ذات بعدين:

two dimensional array ، وتحتوي على 4 × 3 أي 12 عنصراً .

والشكل (1–5–4) يبين برنامجاً لحساب عدد الطلبة الكلي (أي في الأقسام الثلاثة المذكورة) في كل سنة دراسية .

ونلاحظ استخدام الفئة deptset التي تحتوي على الأقسام الثلاثة ، وأن المصفوفة student قد تم التحديد بأنها مصفوفة ARRAY ذات بعدين : الأول من 1 إلى 4 والثاني نطاقه عناصر الفئة debtset .

4–6 المصفوفات المتراصة Packed Arrays

تحتل الثوابت و المتغيرات بأنواعها المتعددة مواقع محددة في الذاكـــرة ، فالمتغير العددي INTEGER يحتل موقعاً متكوناً من 16 خانة ثنائية ، سواء كـــان هذا المتغير يملأ كل هذا الموقع أو بعضاً منه . إلا أن ذلك قد لا يكـــون طريقـــة اقتصادية في التخزين فهو يترك عادة خانات شاغرة غير مستغلة .

وإذا كانت لدينا كمية كبيرة من البيانات يراد تخزينها في الذاكرة فـــإن من المفيد في توفير التخزين أن تشترك بعض المتغيرات في موقع تخزين مشترك . فعلى سبيل المثال قد تكون عناصر المصفوفة أعداداً صحيحة متكونة من خانـــة عشرية واحدة أو اثنتين .

شكل (1–5–4) : برنامج حساب عدد الطلبة في كل سنة دراسية .

```
PROGRAM matrix;
TYPE depts=(computer,stat,math);

VAR dep : depts;
    year : 1..4;
    sum: ARRAY[1..4]OF INTEGER;
    student:ARRAY[1..4, depts]OF INTEGER;

BEGIN
     FOR year := 1 TO 4 DO
     BEGIN
       WRITE('Enter num of students in year',year);
          FOR dep := computer TO math DO
          BEGIN
              IF(dep=computer)THEN
                          WRITE(' Computer-->');
              IF(dep=stat)THEN WRITE(' Stat-->');
```

```
            IF(dep=math)THEN WRITE (' MAth-->');
            READ(student[year,dep]);
        END;
    END;

    FOR year:= 1 TO 4 DO
    BEGIN
        sum[year]:=0;
        FOR dep:=computer TO math DO
          sum[year]:=sum[year]+student[year,dep];
    END;
    FOR year:= 1 TO 4 DO
        WRITELN('total in year ',year, ' is '
                                    ,sum[year]);
END.
```

على الأكثر ، بينما يتسع موقع التخزين مثلاً لأربع خانات عشرية .

في هذه الحالة بالإمكان دمج كل عنصرين من عناصر المصفوفة في موقع واحد مما يقلل من عدد مواقع التخزين المطلوبة . وتوصف هذه المصفوفة في هذا الشكل بأنها متراصة PACKED . ولكن يجب أن نلاحظ هنا أن هذا التوفير في التخزين إنما هو على حساب سرعة معالجة هذه المصفوفة ، حيث يحتاج النظام إلى بعض الوقت عند تفكيك البيانات المتراصة ، و إرجاعها إلى عناصرها المناظرة.

مثال (1–6–4)

إذا كان لدينا عدد كبير من الطلبة ، و كانت المصفوفة PASS عناصرها إما TRUE أو FALSE (أي مصفوفة منطقية) بحيث العنصر I يساوي TRUE إذا كان الطالب I ناجحاً ، ويساوي FALSE إذا كان الطالب راسباً ، يكون من المفيد هنا أن نستخدم المصفوفة PASS على الشكل المتراص ، وذلك على النحو التالي :

```
VAR pass: PACKED ARRAY [1..N] OF BOOLEAN
```

حيث N هو عدد الطلبة . بهذه الجملة يقوم المترجم باستكال تلقائياً باختيار تنظيم مناسب للمصفوفة pass بشكل اقتصادي مناسب يوفر التخزين؛

أي إن هذا التنظيم ليس من مسؤولية المبرمج . علماً بأن هذا التنظيم لن يؤثـــــر على البرنامج ونتائجه ، فهذه النتائج لا تعتمد على كون المصفوفات متراصـــة أو غير متراصة ، ولكن يؤثر استخدام الشكل المتراص في سرعة إنجاز التنفيذ ، كمـــا ذكرنا .

مثال (2–6–4)

كمثال على المصفوفة المتراصة المتكونة من رموز (characters) النـــوع النضيد STRING الذي سبق وأن استخدمناه في البنود السابقة ، حيث تعتـــبر أي مجموعة من الرموز (حروف ، أرقام ، إشارات ،... ، الخ) مثل :

' أحمد عبد السلام'

' interation = '

' x + y = z '

من النوع STRING وهي تكافئ :

PACKED ARRAY [1..N] OF CHAR

حيث N عدد الرموز في النضيد .

وبإمكاننا تحديد النوع names الذي يدل على اسم الشخص إما بالجملة :

TYPE names = STRING [20]

أو بالجملة :

TYPE names = PACKED ARRAY [1..20] OF CHAR

ولكن عند استخدام تربو باسكال نلاحـــظ أن اســـتعمال المصفوفـــة المتراصة من الرموز يتطلب أن يكون عدد الرموز مساوياً تماماً للعدد المعين لهـــا .

فإذا كان المتغير nm من النوع names فإن الجملة :

nm: = ' Mohamed '

تعتبر خطأ (type mismatch) نظراً لعدم وجود 20 حرفاً (رمـزاً) في الاسم . ولكن هذه المشكلة لا تبرز إذا استخدمنا النوع STRING بـدلاً مـن PACKED ARRAY ؛ إذ يقوم المترجم بتكملة الرموز الناقصة بفراغات على يمين الاسم ، وتسمى هذه العملية بالتعديل مـن اليسـار(left- justified) ؛ إذ إن الفراغات توضع على الجهة اليمنى .

7-4 السجلات و الملفات

لقد سبقـت الإشارة إلى أن تطبيقات الحاسب الآلي تحتـوي علـى مجالين أساسيين ، هما : التطبيقات العلمية والتطبيقات الإدارية . وهذا التقسـيم أدى بالتالي إلى اختلاف الأجهزة والتخصصات واللغات في مجال علم الحاسـب. فالتطبيقات العلمية ترتكز على إجراء الكثير من العمليات الحسابية ، ولا تهتـم بمسائل معالجة البيانات الضخمة ، كما هو الحال في التطبيقات الإدارية. ولغـة باسكال هي في الأصل لغة علمية ، ولكنها تحتوي على تعليمـات وإمكانيـات جيدة للتعامل مع البيانات المخزنة على الأشرطة والأقراص .

نبدأ أولاً ببعض التعريفات الأساسية :

1- الوحدة البيانية : (data item) وتعبر عن معلومة واحدة ، مثـل الاسـم أو العمر أو الدخل .

2- السجل (record) وهو مجموعة من الوحدات البيانية ذات علاقة. فالبيانـات عن اسم الشخص مع عمره ودخله تعبر عن سجل يتكون من 3 وحـدات بيانية .

3- الملف (file) وهو مجموعة من البيانات المتناظرة . كأن تكون سجلات عـن طلبة كلية ما ، أو سجلات عن قطع الغيار المتوفر بالمخزن .

لتحديد نوع سجل و مكوناته ، نستخدم جملة TYPE على النحو :

```
TYPE r = RECORD
        f1: typef1;
        f2: typef2;
        ..............
        fn: typefn
END;
```

حيث r هو اسم السجل ، و fn, ... , f3, f2, f1 هي أسماء مكونات السجل على الترتيب ، و typef1 هو نوع f1 ، و typef2 هو نوع f2 ، ... إلخ .

وبعد تحديد النوع r ، يمكننا تحديد المتغير الذي يعبر عن ملف السجلات من النوع r (وليكن اسمه f) على النحو التالي :

```
VAR f: FILE OF r;
```

مثال (1-7-4) :

يبين البرنامج بالشكل (1-7-4) كيفية تكوين ملف نوعي (Typed file) . واسم هذا الملف (كما هو واضح في البرنامج) هو Studentf.dta

يبدأ هذا البرنامج بتحديد تركيبة السجل student في هذا الملف .

ويتحدد اسم السجل ومركباته في جملة TYPE . في هذا البرنامج ويتركب السجل (STUDENT) من 5 مكونات (fields) هي الرقم (number) و الاسم (name) و 3 درجات مختلفة (gr1, gr2, gr3) .

شكل (1–7–4) : برنامج تكوين ملف نوعي على القرص .

```
PROGRAM creatFile;
TYPE student=RECORD
     number : INTEGER;
     name : STRING[20];
     gr1, gr2, gr3 : REAL;
END;
VAR studentFile: FILE OF student;
    rec : student;
    i, num ,n : INTEGER;
    sname : STRING[20];
BEGIN
     ASSIGN(studentFile, 'studentf.dta');
     REWRITE(studentFile);
     WRITE('Enter number of students-->');
     READ(n);
     FOR i:= 1 TO n DO
     BEGIN
          WRITE('Enter number-->');
          READLN(rec.number);
          WRITE('Enter name-->');
          READLN(rec.name);
          WRITE('Enter grade1-->');
          READ(rec.gr1);
          WRITE('Enter second grade2-->');
          READ(rec.gr2);
          WRITE('Enter grade3-->');
          READ(rec.gr3);
          WRITE(studentFile,rec);
     END;

     CLOSE(studentFile);
END.
```

ويتضح الاختلاف بين تركيبة المصفوفة وتركيبة السجل في كون المصفوفة متجانسة العناصر ، أي أن جميعها من النوع نفسه ، بينما قد تختلف مكونات السجل في أنواعها . فبينما الرقم عدد صحيح ، نجد أن الاسم هو نضيد من الحروف (string) بينما الدرجات من النوع الكسري .

نلاحظ في البرنامج استخدام الجمل التالية :

1- جملة ASSIGN لتعيين اسم الملف الذي يتم التخزين به على القرص .

2- جملة REWRITE لفتح ملف جديد للكتابة فيه (أي لتكوينه) .

3- جملة WRITE لكتابة السجلات في الملف (وليس WRITELN).

4- جملة CLOSE لقفل الملف .

ونلاحظ أيضاً أن عدد السجلات التي تسجل في الملف طبقاً لهذا البرنامج هو 5 .

ملاحظة : إذا كان الملف موجوداً ، وتم فتح ملف باستخدام REWRITE فإن البيانات بالملف تضيع .

مثال (2-7-4) :

المطلوب كتابة برنامج لحساب مجموع الدرجات الثلاث في سجل كل طالب وارد في الملف (student.dta) الذي تم تكوينه في المثال (1-7-4) .

يبين الشكل (2-7-4) البرنامج المطلوب ، ونلاحظ في هذا البرنامج مايلي :

1- يشير المتغير total لمجموع الدرجات الثلاثة .

2- استخدام جملة RESET ضروري عند فتح ملف للقراءة منه . ويمكن أيضاً الكتابة فيه .

3- استخدام الدالة المنطقية (البولية) المعروفة باسم EOF (اختصار للكلمـــات End Of File) ، وتعتبر قيمة هذه الدالة خاطئة (FALSE) إلا إذا تم الوصول إلى آخر سجل في الملف . أي أن الجملة :

WHILE NOT EOF (studentfile) DO ..

تعني :

طالما لم تصل إلى **نهاية الملف** (studentfile) أنجز ..

4- للقراءة من الملف ، نستخدم جملة READ كما في الجملة :

READ (studentfile, rec)

حيث rec متغير يرمز لسجل الملف .

5- استخدام جملة CLOSE لقفل الملف عند الانتهاء من تكوينه .

شكل(2-7-4) : برنامج لتوضيح القراءة من ملف نوعي .

```
PROGRAM readFile;
TYPE student= RECORD
     number : INTEGER;
     name : STRING[20];
     gr1, gr2, gr3 : REAL;

END;
VAR studentFile : FILE OF student;
    rec : student;
    total : REAL;

BEGIN

      ASSIGN(studentFile,'studentf.dta');
      RESET(studentFile);
      WHILE NOT EOF(studentFile) DO
      BFGIN

           READ(studentFile,rec);
           total := rec.gr1 +  rec.gr2 +
```

```
                                        rec.gr3;
            WRITELN(rec.number,

                '    ',rec.name,'    ',total:6:2);
        END;
END.
```

4-8 الملفات النصية Text Files

سبق وأن أشرنا إلى أن الملف هو مجموعة من البيانات المتناظرة ، أي أنه يتكون من تركيبة بيانات معينة ، كأن يكون مجموعة من السجلات . وهذا التعريف هو السائد في أغلب لغات البرمجة ، إلا أننا هنا سنشير إلى هذا النوع بأنه ملف نوعي (typed) ، وذلك تمييزاً له علما يسمى بالملف النصي (text file) ، وهو الملف الذي يتكون من البيانات التي لا تتبع نسقاً معيناً. فمثلاً يعتبر هذا الكتاب ملفاً نصيًّا إذ إن محتوياته كلها رموز لا تخضع لنسق ثابت ، كما يمكن اعتبار أي برنامج مكتوب بإحدى لغات البرمجة على أنه ملف نصي.

وبالتالي ليس هناك ضرورة لتحديد نوع مكونات الملف النصي ، بل يكفي تحديد نوع الملف (وليكن اسمه مثلا f) على أنه من النوع النصي في جملة VAR على النحو :

VAR f: TEXT;

مثال (1-8-4)

المطلوب كتابة برنامج لتخزين ملف نصي بعد إدخال بياناته من لوحة المفاتيح (Keyboard) ، مع قفل الملف عند إدخال الرمز '@' .

يبين الشكل (1-8-4) البرنامج المطلوب . نلاحظ وجود 3 متغيرات في البرنامج : المتغير الأول اسمه c من النوع الرمزي ، وهو ينقل الرمز الواحد من لوحة المفاتيح إلى الملف المطلوب إلا إذا كان الرمز هو @ عندها يتم الإدخال وقفل الملف . أما المتغير t فهو يقوم بتعريف الملف النصي ، ونستخدم المتغير f1 لتعين اسم للملف في جملة ASSIGN ، ويتم تخزينه بهذا الاسم على القرص. لاحظ أن هذا الاسم يتكون من 8 حروف على الأكثر إضافة إلى 3 حروف في امتداده .

شكل (1-8-4) : برنامج لتكوين ملف نصي .

```
PROGRAM CreatText;
VAR c: CHAR;
    t: TEXT;
    f : STRING[12];
BEGIN

     WRITE('Enter file name-->');
     READLN(f);
     ASSIGN(t,f);
     REWRITE(t);
     WRITELN('Enter your text:');
     READ(c);
     WHILE c<>'@' DO
     BEGIN

          WRITE(t,c);
          READ(c);
     END;

     CLOSE(t);
END.
```

9-4 تمـــــــارين

1- إذا كان المتغير الصحيح YEAR يرمز للسنوات مـن 1960 إلى 2000 ، فكيف نحدد ذلك في لغة باسكال ؟

2- ما الخطأ في الجملة التالية ؟

```
TYPE marks = ( 55, 77, 85, 90);
```

3- ما الخطأ اللغوي في البرنامج التالي ؟

```
PROGRAM  grades;
TYPE gr = ( a, b, c, d, f, w);
VAR g: gr;
BEGIN
READLN (g);
IF ( g = f ) OR ( g = w) THEN
WRITELN ('FAIL')
ELSE WRITELN ('PASS')
END.
```

4- حاول استنتاج الغرض من البرنامج في التمرين (3) وأعد كتابة البرنـــامج بحيث يؤدي هذا الغرض .

5- اكتب برنامجـــاً يطبع عدد الأيام في أشهر الســـنة الميلاديـــة الكبيســـة مستخدماً الجملة :

```
TYPE month = ( jan, feb, mar, apr, may, jun, jul, aug, sep,
               Oct, nov, dec )
```

6- استخدم مصفـــوفات أحادية البعد لقراءة مجموعة من الأعداد الكســـربة ثم حساب :

- متوسط هذه الأعداد .

- مجموع مربعات الفروق بين كل عدد مقروء والمتوسط .

7- أعد كتابة البرنامج في المثال (1-4-4) بحيث يصبح الترتيب تصاعدياً. افترض أن الأرقام كسرية موجبة وأنها لا تزيد عن 100 رقم .

8- اكتب برنامج لترتيب قائمة تحتوي على الاسم والرقم بحيث يكون الترتيب بناء على الرقم .

9- اكتب برنامجاً لطباعة جميع الكلمات التي يمكن تكوينها من 4 حروف معطيات .

10- إذا كانت لديك مصفوفة عناصرها كسرية ، وتتكون من 5 صفوف و 6 أعمدة ، فاكتب :

- برنامجاً لحساب مجموع عناصر كل صف .
- برنامجاً لحساب عناصر كل عمود .
- برنامجاً لحساب مجموع العناصر التي يتساوي فيها رقم الصف والعمود .

11- اكتب برنامجاً لحساب وطباعة جدول الرواتب كما يلي :

العمود الأول يحتوي على أرقام الموظفين .

العمود الثاني يحتوي على أسمائهم .

العمود الثالث خاص بالمرتب الأساسي .

العمود الرابع خاص بالعلاوات .

العمود الخامس خاص بالخصميات .

العمود السادس لإجمالي المرتب .

علماً بأن الأعمدة من 1 إلى 5 تعتبر من المعطيات ، وأن إجمالي المرتب = المرتب الأساسي + العلاوات – الخصميات . استخدم المصفوفات ذات البعد الواحد والبعدين في البرنامج . افترض أن لديك معلومات عن 5 موظفين في إجراء البرنامج ، يتم إدخالها عن طريق لوحة المفاتيح (Keyboard) .

12- من الناحية العلمية ، لا يتم إدخال البيانات عن طريق لوحة المفاتيح كلما أجرينا البرنامج ، ولكنها تخزن في ملف ثم يستدعي هذا الملف من قبل البرنامج . اكتب برنامجاً لتكوبن ملف للبيانات في تمرين (11) وتخزينه على القرص .

13- استخدم الملف الذي كونته في تمرين (10) لحساب وطباعة إجمالي المرتب ، كما هو مبين بذلك التمرين .

14- اكتب برنامجاً لعرض أي ملف نصي على الشاشة ، بحيث يتم إدخال اسم الملف من لوحة المفاتيح .

* * *

4-10 الفئات SETS

إلى جانب تركيبة المصفوفة ، وتركيبة السجل ، يوجد نوع آخر من تراكيب البيانات في لغة باسكال وهو الفئة . والفئة تشبه المصفوفة من حيث كونها مجموعة من الأشياء ذات نوع واحد وتحت اسم واحد ، ولكن هناك فرقاً جوهريًّا بين الاثنين ، يتلخص في أن عنصر المصفوفة له دليل يبين موقعه من بين العناصر الأخرى ، وهي خاصية لا تتوفر للفئة . وفي المقابل تتمتع الفئات

بخواص، مثل : الفئة الجزئية ، وإيجاد التقاطع والاتحاد ، إلى غير ذلك من العمليـــات المألوفة.

نبدأ بكيـــفية تحديد متغير بأنه يصف فئة . أولاً يجب تحديد الفئة الشـــاملة التي ينتمي إليها. فإذا كـــانت الفئة summer تعبر مثلاً على أشهر الصيف:

(Jun, Jul, Aug)

وكانت الفئة الشاملة هي أشهر السنة :

month = (Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep,
Oct, Nov, Dec)

فيمكننا تحديد ذلك على النحو المبين في الشكل (1-9-4) .

شكل (1-9-4) : برنامج لتوضيح استخدام الفئات .

```
PROGRAM sets;
TYPE
monthsOfYear=(jan,feb,mar,apr,may,jun,jul,
                        aug,sep,nov,dec);
VAR m : monthsOfYear;
    summer : SET OF MonthsOfYear;
BEGIN
     summer:=[jun, jul, aug ];
     FOR m:= jan TO dec DO
          IF m IN summer THEN
             WRITELN('warm')
          ELSE
              WRITELN('cold');
END.
```

ونلاحظ في هذا البرنامج ما يلي :

1- استخدام المصطلح SET OF في تحديد المتغير summer بأنه فئة جزئيـــة من الفئة الشاملة month .

2- استخدام الكلمة IN للتعبير عن إنتماء عنصر لفئة معينـــة ؛ أي إن (m IN summer) تعني أن m تنتمي إلى الفئة summer .

ويمكننا إجراء بعض العمليات على الفئات كما يلي :

A * B تقاطع (Intersection) الفئة A مع B

A + B اتحاد (Union) الفئتين

A – B فئة العناصر في A التي لا تنتمي إلى B .

كما نلاحظ هنا أن الفئة الخالية نرمز لها بالرمز [ ].

والبرنامج المبين في الشكل (2-9-4) يوضح هذه العمليات على الفئات التالية (و هي تمثل أسماء الطلبة المسجلين بكل مقرر) :

Basic: = [Sami, Nuri, Lila, Huda];

Pascal: = [Nuri, Jamila, Huda];

والمطلوب إيجاد عدد الـــطلبة المسجلين في كلا المقررين (أي التقـــاطع) والمسجلين في أحد المقررين (أي الاتحاد union) والمسجلين في Basic ولكـــن غير مسجـــلين في Pascal ، ونرمز لهم بالفئة bmp . لاحظ أن I عدّاد التقاطع و k عداد الاتحاد و l عداد الفئة bmp .

شكل(2-9-4) : برنامج عمليات الفئات .

```
PROGRAM SetOperations;
TYPE
    students=(Ali, Huda,Jamila,Lila, Nuri,
                  Salwa, Sami);
VAR
    s : students;
```

```
    i , k, L : INTEGER;
    basic, pascal, intersection, union, bmp:
                              SET OF students;
BEGIN
     basic := [Sami, Nuri, Lila, Huda];
     Pascal := [Nuri, Jamila, Huda];
     intersection := basic*pascal;
     union := basic + pascal;
     bmp := basic - pascal;
     i:=0;   k:=0;   L:=0;
     FOR s:= Ali TO Sami DO
     BEGIN
          IF s IN intersection THEN i:=i+1;
          IF s IN union THEN k:=k+1;
          IF s IN bmp THEN L:=L+1;
     END;
    WRITELN(' Students registered in both BASIC
                         and PASCAL is ',i);
     WRITELN(' Students registered in BASIC or
                              PASCAL is ',k);
    WRITELN(' Students registered in BASIC and
                        NOT in PASCAL is ',L);
END.
```

* * *

4-11 تمـــــارين

1- المطلوب كتابة برنامج لإعداد جدول ، وذلك بإيجاد الفئة المشتركة بـين مجموعة من الفئات ، حيث لدينا عدد من الأشـــخاص (وليكـــن 5) ولكـــل شخص فئة تبين الأيام الشاغرة لديه (وليكن عددها 3) . اكتب هذا البرنـــامج بحيث :

- يقوم بقراءة فئة كل شخص علماً بأن الفئة الشاملة هي [7 ... 1] .
- إذا كان تقاطع الفئات المدخلة فئة خالية يطبع كلمة (empty) وإلاّ فيطبـع عناصر هذه الفئة .

2- أعد كتابة البرنامج في تمرين (1) بحيث يكـــون صالحـــاً لأي عـــدد مـــن الأشخاص، وبحيث تتم القراءة من ملف نصي .

3- في مؤسسة ما يوجد 20 موظفاً ، لكل موظف رئيـــس مباشـــر . اكتـــب باستخدام الفئات برنامجاً يقوم بقراءة رقم كل موظـــف (مـــن 1 إلى 20) إلى جانب رقم رئيسه المباشر ، ثم يطبع كل رئيس والموظفين التابعين له . افـــترض وجود 4 رؤساء أقسام .

* * *