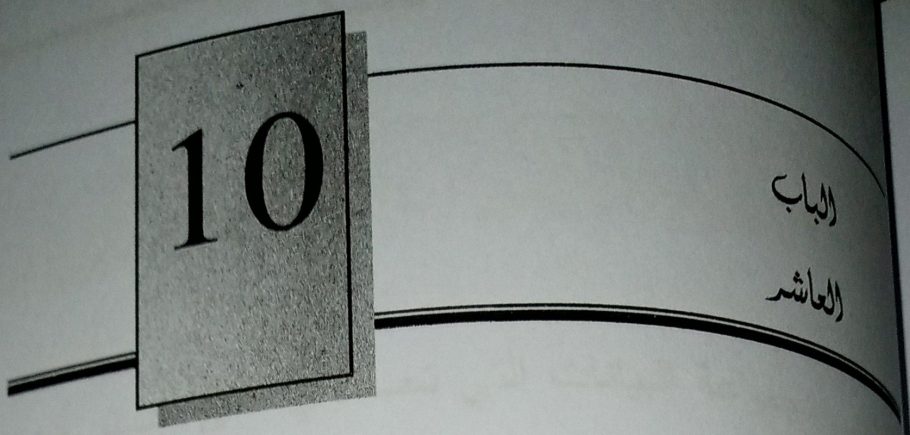
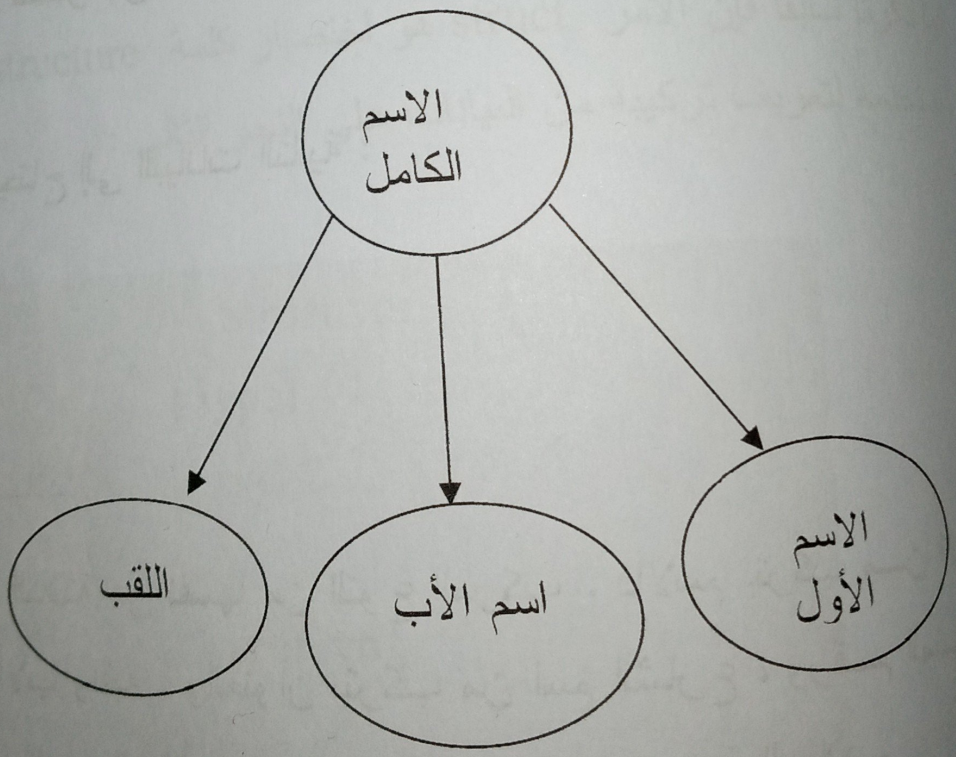


تركيبة سجل البيانات

10.1 مقدمة

تسمى مجموعة البيانات التي تحتوي على أكثر من عنصر (مع إمكانية اختلاف أنواع عناصرها) بالتركيبة structure . ولعل أبسط مثال على تركيبة بيانات هي اسم الشخص ، حيث يتركب الاسم عادة من ثلاثة عناصر : الاسم الأول ، واسم الأب ، واللقب . ويمكن أن نمثل هذه التركيبة بالشكل الآتي :



نلاحظ هنا أن العناصر الثلاثة للاسم هي من نوع واحد وهو النضيد string. ولكن ذلك غير ضروري ، فقد تختلف أنواع العناصر في البيانات المركبة . خذ على سبيل المثال بيانات مقرر دراسي بقسم الحاسوب ، فهذه البيانات قد تحتوي

v1 , v2 , … , vn

ونوع هذه الفروع هو :

type1 ، type2 ، …. typen

على التوالي .

فمثلا التركيبة :

```
Struct name
   {  char first[20] ;
      char last[20] ;
   } ;
```

اسمها name ، وتتكون من فرعيين هما first و last ، وكلاهما من النوع النضيد string . لاحظ ضرورة وضع القوسين { } بعد اسم التركيبة ، وضرورة وضع فاصلة منقوطة بعد القوسين .

الآن وبعد تعريف التركيبة name يمكننا تعريف أي متغير بأنه من النوع name وذلك على الصورة التالية :

struct name nm ;

وهذا يعني أن المتغير nm هو من النوع name، أي أنه يتكون من فرعين هما first و last. بعد هذا التعريف للمتغير nm نستطيع أن نشير إلى الفرع first في البرنامج بالمتغير:

الحديثة ، ولكن قد يختلف اسم الموضوع من لغة إلى أخرى . فبينما نرمز له في لغة سي بالرمز struct (اختصارا لكلمة structure أي تركيبة) ، نرمز له في لغـــة أخرى مثل باسكال بالرمـــز record أي سجل .

10.2 تحديد تركيبة بالأمر struct

كما ذكرنا سابقا فإن الأمر struct هو اختصار كلمة structure أي تركيـــة ، ونستخدمه لتعريف تركيبة من البيانات على النحو التالي :

```
Struct var
{ type1 v1 ;
  type2 v2 ;
     ...
  typen vn ;
} ;
```

ومعنى ذلك أن اسم التركيبة (السجل) هو var ، وأن هذه التركيبة تتكون من العناصر (الفروع) التالية:

على الآتي:

1 . اسم المقرر (نضيد) .

2 . رمز المقرر (عددي أو نضيد) .

3 . محتوي المقرر (نضيد) .

4 . عدد الطلبة المسجلين بالمقرر (عددي صحيح) .

5 . عدد الساعات الدراسية أسبوعياً (عددي صحيح) .

وكمثال آخر ، لننظر إلى سجل طالب بالجامعة. ما هي البيانات التي نريد معرفتها عنه؟

على الأقل نحتاج إلى البيانات التالية :

1 . الاسم

2 . العنوان

3 . تاريخ الميلاد

هذه العناصر الثلاثة هي نفسها من النوع المركب . فالاسم يتركب من الاسم الأول واسم الأب ولقبه . والعنوان يتركب من اسم الشارع ، ورقم المنزل ، والمدينة . أما تاريخ الميلاد فيتركب من اليوم والشهر وسنة الميلاد .

معني ذلك أن عناصر نوع مركب قد تكون هي نفسها مركبة .

إن موضوع تراكيب البيانات ليس خاصاً بلغة سي ، بل تشترك فيه معظم اللغات

```
        age.day = 30 + today.day - birth_day.day;
        today.month-- ;
    }
    if(birth_day.month < today.month )
        age.month=  today.month - birth_day.month;
    else
    {
        age.month = 12 + today.month - birth_day.month;
        today.year-- ;
    }
    age.year=today.year-birth_day.year;

    printf("\n Your age is %d years, %d months and %d
days",      age.year, age.month, age.day);
}
```

الشكل (10.2.1) برنامج حساب العمر

يظهر في البرنامج (10.2.1) وجود ثلاثة متغيرات من النوع المركب date :

age , birth_day , today

يتم تعريفها بالجمل التالية :

```
struct date today
struct date birth_day ;
struct date age ;
```

Your age is 58 years , 11 months , and 5 days .

```
main()
{
    struct date
    {
        int day;
        int month;
        int year;
    };
    struct date today, birth_day, age;
    printf("\n enter today's date");
    printf("\n day? ");
    scanf("%d",&today.day);
    printf("\n month? ");
    scanf("%d",&today.month);
    printf("\n year? ");
    scanf("%d",&today.year);
    printf("\n\n enter birthday \n day? ");
    scanf("%d",&birth_day.day);
    printf("\n month? ");
    scanf("%d",&birth_day.month);
    printf("\n year? ");
    scanf("%d",&birth_day.year);

    if(birth_day .day < today.day)
        age.day=today.day - birth_day.day;
    else
    {
```

nm.first

ونشير إلى الفرع الثاني بالمتغير :

nm.last

على سبيل المثال ، نستطيع أن نكتب جملة مثل

strcpy (nm.last , " Omar Zarty ") ;

مثال (10.2.1) : اكتب برنامجا لحساب العمر age ، وذلك بحساب الفرق بين تاريخ اليوم today وتاريخ الميلاد birthDay ، علما بأن كليهما من النوع date حيث يتكون من اليوم/الشهر/ السنة.

لتبسيط الحساب افترض أن الشهر = 30 يوما .

يبين الشكل (10.2.1) البرنامج المطلوب الذي يتم تنفيذه على النحو التالي :

```
enter today  date:
        day ? 26
        month ? 8
        year ? 2005
enter birth day:
        day ? 21
        month ? 9
        year ? 1946
```

ثم لتـــعريف متغير ما مثل person بأنه من النوع record نكتب الجملة :

struct record person ;

والمثال التالي يجمع هذه الأمثلة في برنامج واحد .

مثال (10.3.1) : اكتب برنامجا لـــقـــراءة اسم الطالب وعنوانه وطباعة الآتي :

Mr ..... lives in ..... , ... , ...

حيث تملأ الفراغات بالاسم الكامل والعنوان الكامل .

من المهم أن نلاحظ في هذا البرنامج استخدام النقطة فـــي فـــصل الجـــد والأب والابن . على سبيل المثال في المتغير

person.adrs.street

نبدأ بالجد على اليسار (person) ثم الأب adrs ثم الابن street.

يتم تنفيذ البرنامج (10.3.1) كما في المثال التالي :

enter first name → Omar
enter last name → Zarty
enter street → Kafaja

على الاسم والعنوان ، وكلاهما من النوع المركب حيث الاسم يحتوى على الأقل على الاسم الأول واللقب ، أما العنوان فيتكون من الرقم والشارع والمدينة .

في هـذه الحالة نقوم أولا بتعريف التركيبات الأولية ، أي فـي هـذا المثـال ، تركيبة الاسم والعنوان ، مثل :

```
Struct  name
 {  char first[10] ;
    char last[10] ;
} ;

struct address
{
char num[4] ;
char street[12] ;
char city[20] ;
} ;
```

والآن نستطيع تعريف التركيبة record لسجل الطالب على النحو :

```
struct record                    {
        struct name nm ;
        struct address adrs ;
    } ;
```

ولكن يمكن اختصار ذلك كما في البرنامج ، في جملة واحدة :

```
struct date today , birth_day , age ;
```

وعند التعامل مع فرع من هذه التراكيب نستخدم النقطة dot للفصل بين التركيبة الأم والابن مثل :

```
today.day
```

أو

```
age.month
```

وهكذا .

لاحظ أيضا في منطق البرنامج عملية (الاستلاف) عند إجراء طرح تاريخ الميلاد من تاريخ اليوم ، وهذا ما يفسر إضافة 30 يوما ، وطرح 1 من الشهر ، في حالة أن يوم الميلاد أكبر من رقم اليوم الحالي . وكذلك الحال عندما يكون شهر الميلاد أكبر من رقم الشهر الحالي ، حيث تضيف 12 إلى الشهر وتطرح 1 من السنة .

10.3 تركيبة ذات عناصر مركبة

كثيراً ما نواجه تراكيب ذات عناصر هي نفسها مركبة ، وأبسط مثال على ذلك – كما أشرنا في المقدمة – سجل موظف أو طالب . فهو يحتوى على الأقل

فمثلا الإعلان التالي :

```
struct cloths
{    char size ;
float price ;
      }
shirts , pants  ;
```

يقوم بتعريف النوع cloths ، وأيضاً بتعريف المتغيرين shirts و pants من النوع cloths .

والإعلان التالي :

```
struct grades
  { int num ;
    float gr ;
  }
     student[3] =
      {    101 , 59.6
           102 , 86.7
           103 , 70.5
       } ;
```

```
printf("\n enter number→");
gets(person.adrs.num);
printf("\n enter city→");
gets(person.adrs.city);
printf("\n Mr. %s %s lives in %s, %s Street, %s ",
        person.nm.first,
       person.nm.last, person.adrs.num,
        person.adrs.street,
       person.adrs.city);

}
```

الشكل (10.3.1) برنامج لتركيبة عناصرها مركبة .

وينتج عن ذلك طباعة الرسالة التالية على الشاشة :

Mr Omar Zarty lives in 32 , Kafaja street , Tripoli

10.4 جدول البحث lookup table

لاحظ عند إعلان أي تركيبة ضرورة وضع الفاصلة المنقوطة بعد القوسين { } ، والسبب في ذلك إمكانية الإعلان عن متغير أو مصفوفة من نوع هذه التركيبة (تسمي بجدول البحث lookup table) بعد الأقواس وقبل الفاصلة المنقوطة .

enter number → 32
enter city → Tripoli

```
#include <string.h>
main()
{
        struct name
        {
                char first[10];
                char last[10];
        };
        struct address
        {
                char num[4];
                char street[12];
                char city[15];
        };
        struct record
        {
                struct name nm;
                struct address adrs;
        };
        struct record person;
        printf("\n enter first name→");
        gets(person.nm.first);
        printf("\n enter last name→");
        gets(person.nm.last);
        printf("\n enter street→");
        gets(person.adrs.street);
```

```
    for(k=0; k<NS; k++)
        if(student[k].gr > ave )
            printf("\n %d %6.2f", student[k].num,
student[k].gr);
}
```

الشكل (10.4.1) استخدام جدول البحث في التركيبة

ومن الملاحظ في البرنامج (10.4.1) أن المصفوفة student عناصرها مركبة ، حيث تتركب من الرقم والدرجة . في هذه الحالة نرمز لرقم الطالب k بالصورة التالية :

student[k].num

وليس student . num[k] كما قد يتبادر إلى الذهن . وكذلك بنفس الطريقة نرمز لدرجة الطالب k، أي :

student[k].gr

وملاحظة أخيرة عن البرنامج (10.4.1) أنه يمكن كتابته بدون استخدام جدول البحث ، وذلك بقراءة البيانات بواسطة الدالة scanf ، مع إعلان المصفوفة student بأنها من النوع grades كالآتي :

struct grades student[NS] ;

```
#define NS 10
main()
{
    struct grades
    {
        int num;
        float gr;
    }
    student[NS]=
    {
        101, 59.6,
        102, 67.8,
        103, 70.5,
        104, 44.5,
        105, 55.5,
        106, 77.5,
        107, 82.0,
        108, 54.0,
        109, 66.5,
        110, 90.0
    };
    int k;
    float ave, sum=0;
    for(k=0; k<NS; k++)
        sum += student[k].gr;
    ave= sum/NS;
    printf("\n average=%6.2f", ave);
    printf("\n\n list of above average student numbers:");
```

يقوم بتعريف التركيبة grades من جزءين : الرقم num والدرجة gr . في نفس الوقت يتم تعريف المصفوفة student من 3 عناصر من النوع grades وتعطي قائمة بقيم هذه العناصر ، واختصرنا بذلك كتابة الجمل التالية :

```
student[0].num = 101 ;
student[0].gr = 59.6 ;
student[1].num = 102 ;
student[1].gr = 67,8 ;
student[2].num = 103 ;
student[3].gr = 70.5 ;
```

مثال (10.4.1) :

اكتب برنامجا يقوم بإدخال أرقام ودرجات 10 طلبة في جدول ، بحيث يتم عرض أرقام الطلبة الذين تحصلوا على درجة فوق المتوسط .

يبين الشكل(10.4.1) البرنامج المطلوب.

لاحظ أن عملية سرد القائمة عموديا ، بحيث يظهر كل سجل في سطر واحد ، هي لغرض التوضيح ولكن ليست ضرورية ، أي يمكننا سرد البيانات أفقيا في سطر واحد أو أكثر ، مع ضرورة فصلها عن بعضها بالفاصلة العادية .

مثال (10.5.1) : اكتب الدالة cal_time التي تحسب عدد الثواني والدقائق والساعات في زمن t من الثواني .

لاحظ أنه بإمكاننا كتابة هذه الدالة باستخدام المؤشرات والاستدعاء بالعنوان call-by-address ، ولكننا هنا نريد أن نستغني عن استخدام المؤشرات ، وأن نقوم بترجيع القيم المطلوبة (دفعة واحدة) أي في

```
typedef struct
{
        int sec;
        int min;
        int hrs;
}time;
main()
{
        long seconds;
        time t;
        time cal_time( long s );
        printf("\n enter time in seconds->");
        scanf("%ld",&seconds);
        t= cal_time(seconds);
        printf("\n hours=%d  minutes=%d  seconds=%d",t.hrs,
t.min, t.sec);
}
time cal_time (long s)
{
```

```
struct time compute ( struct time t1 , struct time t2 ) ;
```

لتعريف دالة اسمها compute (وهي من النوع time الذي تم تعريفه) ، وهي تستقبل متغيرين t1 و t2 من النوع time أيضاً .

ونظراً لوجود تكرار لعبارة struct time هنا ، نستخدم جملة تعريف النوع typedef (اختصار type definition) التي تقوم بتعريف نوع جديد مثل time بالصورة التالية :

```
typedef struct
{    int sec ;
     int min ;
     int hrs ;
} time ;
```

بهذا نكون قد عرفنا نوعاً جديداً اسمه time ، ويمكن تعريف أي متغير v بأنه من هذا النوع بالطريقة البسيطة :

```
time  v ;
```

وبهذا يمكننا إعلان الدالة compute بالطريقة التالية :

```
time compute ( time t1 , time t2 );
```

ولكن الغرض من البرنامج هو توضيح استخدام جدول البحث الذي نضطر إلى استخدامه خاصة عندما نريد أن تظهر البيانات في البرنامج ، أي عندما تكون البيانات جزءاً من البرنامج ، أو عندما نريد أن نعطي للمصفوفة قيما ابتدائية initial values .

10.5 الدالة من النوع المركب

كما أن هناك متغيرات من النوع المركب ، يمكن أن تكون الدالة أيضاً من النوع المركب . خذ مثلا الدالة compute التي تحسب الزمن بالساعات والدقائق والثواني. إذا عرفنا هذه التركيبة على النحو :

```
struct time
  { int sec ;
    int min ;
   int hrs ;
} ;
```

يمكننا استخدام الإعلان :

أما عملية الطرح فتتم على النحو التالي :

$$\frac{a}{b} - \frac{c}{d} = \frac{ad - bc}{bd}$$

أما عملية الضرب فهي ببساطة

$$\frac{a}{b} * \frac{c}{d} = \frac{ac}{bc}$$

وأخيراً فإن عملية القسمة تكون على الشكل التالي :

$$\frac{a}{b} \div \frac{c}{d} = \frac{ad}{bc}$$

والآن نقوم بكتابة دالة لكل عملية حسابية . ويبين الشكل (10.5.2) برنامجاً كاملاً يحتوي على هذه الدوال الأربع ، وقد تم تنفيذه وأعطى النتائج التالية :

1/3 + 1/ 4 = 7/12
5/6 - 2/3 = 3 /18
7/8 * 3/18 = 21 / 144
result = 7/12 / 21 / 144 = 1008 / 252

hours = 1 minutes = 32 seconds = 35

وللتأكد من صحة هذه الإجابة ، نحسب الثواني المقابلة لها :

1 hr + 32 min + 35 sec = 1* 60* 60 + 32* 60 + 35 = 5555

لاحظ أننا استفدنا من مؤثر باقي القسمة % في كتابة الدالة ، فمثلا إذا كان الزمن بالثواني هو s فإن باقي قسمة s على 60 هو جزء الثواني من التركيبة time .

مثال (10.5.2) : يتكون الكسر الاعتيادي من جزءين صحيحين هما البسط numerator والمقام denominator . اكتب الدوال mul_r , div_r لعملية جمع، وطرح ، وضرب ، وقسمة كسرين اعتياديين . واستخدمها لحساب الكسر الاعتيادي الناتج من قسمة $\frac{1}{3}+\frac{1}{4}$ على $\left(\frac{5}{6}-\frac{2}{3}\right)*\frac{7}{8}$.

لاحظ عند جمع الكسرين $\frac{a}{b}+\frac{c}{d}$ أن الناتج هو الكسر

$$\frac{a}{b}+\frac{c}{d}=\frac{ad+bc}{bd}$$

```
    time t;
    t.sec=s%60;
    t.min=(s/60)%60;
    t.hrs=s/3600;
    return(t);
}
```

الشكل (10.5.1) دالة من النوع المركب

تركيبة واحدة هي التركيبة time التي تتكون من 3 أعضاء هي الثواني والدقائق والساعات .

ملاحظات :

1) يمكننا استخدام جملة typedef لتعريف أي نوع لأي متغير أو دالة .

2) في حالة تعريف دالة من نوع مركب ، يجب أن يكون التعريف عاماً global أي خارج الدوال ، وبهذا وضعنا في البرنامج (10.5.1) تعريف النوع time قبل الدالة الرئيسية main .

3) لاحظ بساطة ترجيع عدة قيم في تركيبة واحدة بالجملة :

return (t) ;

4) تم تنفيذ هذا البرنامج على النحو التالي :

enter time in seconds → 5555

فكانت الإجابة :

```
        int a , b , c, d;
        a= r1.num;
        b= r1.den;
        c= r2.num;
        d= r2.den;
        r.num= a*d - b*c ;
        r.den= b*d;
        return(r);
}
ratio mul_r( ratio r1, ratio r2 )
{
        ratio r;
        int a , b , c, d;
        a= r1.num;
        b = r1.den;
        c= r2.num;
        d = r2.den;
        r.num = a*c;
        r.den = b*d;
        return(r);
}
ratio div_r( ratio r1, ratio r2)
{
        ratio r;
        int a , b , c, d;
        a= r1.num;
        b= r1.den;
        c= r2.num;
        d= r2.den;
```

```
        printf("\n %d/%d - %d/%d = %d/%d",r3.num, r3.den,
r4.num,
                r4.den, r3minusr4.num, r3minusr4.den);
        r5mul = mul_r(r5,r3minusr4);
        printf("\n %d/%d * %d/%d = %d/%d ", r5.num,
r5.den,r3minusr4.num,
                r3minusr4.den,r5mul.num, r5mul.den);
        result = div_r(r1plusr2,r5mul);
        printf("\n result=%d/%d / %d/%d = %d/%d
",r1plusr2.num,
                r1plusr2.den, r5mul.num, r5mul.den, result.num,
result.den);
}
ratio add_r( ratio r1, ratio r2)
{
        ratio r;
        int a, b, c, d;
        a=r1.num;
        b=r1.den;
        c=r2.num;
        d=r2.den;
        r.num=  a*d+ b*c ;
        r.den= b*d;
        return(r);
}

ratio sub_r( ratio r1 , ratio r2)
{
        ratio r;
```

```
typedef struct
{       int num;
        int den;
} ratio;

main()
{
        ratio r1,r2,r3,r4,r5,
            r1plusr2 , r3minusr4, r5mul, result,
            add_r(ratio r1 , ratio r2),
            sub_r(ratio r1, ratio r2),
            mul_r(ratio r1, ratio r2),
            div_r(ratio r1, ratio r2);
        r1.num=1;
        r1.den=3;
        r2.num=1;
        r2.den=4;
        r3.num=5;
        r3.den=6;
        r4.num=2;
        r4.den=3;
        r5.num=7;
        r5.den=8;
        r1plusr2 = add_r(r1,r2);
        printf("\n %d/%d + %d/%d= %d/%d",r1.num, r1.den,
r2.num,
                r2.den, r1plusr2.num, r1plusr2.den);
        r3minusr4 = sub_r(r3,r4);
```


(10.5 برنامجاً

ئج التالية :

```
    d = r2.den ;
    r3pt->num = a*d + b*c;
    r3pt->den = b*d;
    return;
}
```

الشكل (10.6.1) دالة حساب مجموع كسرين اعتياديين باستخدام المؤشرات

ملاحظات عن البرنامج (10.6.1) :

1 . كما سبق و أن ذكرنا فإن إعلان النوع المركب ratio يجب أن يتم خارج الدالة main حتى يكون عاماً لبقية الدوال .

2 . عند تعيين قيم لعضو التركيبة ratio باستخدام المؤشر استخدمنا السهم → بدلاً من النقطة dot في الحالة العادية ، أي :

r3pt → num = a □ d + b □ c ;

r3pt → den = b □ d ;

حيث r3pt هو مؤشر للمتغير المركب r3 من النوع ratio . بصورة عامة ، فالعبارة

vp□ → elem

تعني العنصر elem من التركيبة التي يشير إليها المؤشر vp .

لاحظ أن السهم → هو عبارة عن رمزين متتاليين : الشرطة (علامة الطرح) والرمز > (علامة أكبر من)

مثال (10.6.1) : اكتب الدالة التي تستقبل r1 و r2 مـن النوع ratio (أي النوع المركب من البسط و المقام) ، وترجع r3 حاصل الجمع وهو أيضاً من النوع ratio ، واختبرها بحساب:

$$3/4 \quad + \quad 4/5$$

```
typedef struct
        {       int num;
                int den;
        }ratio;
main()
{
        ratio r1, r2, r3;
        void add_r(ratio r1, ratio r2, ratio *r3pt);
        r1.num = 3;
        r1.den = 4;
        r2.num = 4;
        r2.den = 5;
        add_r( r1, r2, &r3);
        printf ("\n %d/%d + %d/%d = %d/%d ", r1.num, r1.den,
                r2.num, r2.den, r3.num, r3.den);
}
void add_r( ratio r1 , ratio r2, ratio *r3pt)
{
        int a, b ,c ,d;
        a= r1.num;
        b= r1.den;
        c= r2.num;
```

```
    r.num= a*d;
    r.den= b*c;
    return(r);
}
```

الشكل (10.5.2) برنامج إجراء العمليات الحسابية على الكسور الاعتيادية

10.6 مؤشرات النوع المركب

تحتاج أحياناً إلى استخدام مؤشر لنوع مركب ، وهذا يحدث بصورة خاصة عند الاستدعاء بالعنوان . كمثال على ذلك ، التركيبة ratio في المثال السابق والتي ترمز لنوع الكسر الاعتيادي المركب من بسط ومقام . لو أردنا كتابة دالة لجمع عددين من هذا النوع هما r1 و r2 وترجيع حاصل الجمع في المتغير r3 ، فلابد من استخدام عنوان r3 على الصورة :

void add_r (ratio r1 , ratio r2 , ratio *r3)

حيث r3 مؤشر للنوع r3pt .

وعند استدعاء هذه الدالة ، نكتب

add_r (r1 , r2 , &r3) ;

كما في المثال التالي :

من النوع المعدود فإن القيمة الافتراضية لأول عنصر هي الصفر ، يليها الواحد ، وهكذا إلى آخر عنصر .

لاحظ أنه بإمكاننا تغيير القيمة الافتراضية الابتدائية من 0 إلى عدد آخر . فمثلاً إذا أردنا أن يبدأ العد من 1 (وليس من 0) نكتب الإعلان على النحو التالي :

```
enum set { FALL = 1 , WINTER , SPRING , SUMMER };
```

10.8 الاتحاد union

الكلمة المحجوزة union (وتعني لغويا الاتحاد) تستخدم في لغة سي لنفس غرض struct ، أي لإنشاء تركيبة بيانات من عناصر مختلفة ، ولكن هناك فرق أساسي بين الكلمتين وهو أن عناصر التركيبة union تشترك في نفس موقع التخزين في الذاكرة. هذا يؤدي بالطبع إلى أن آخر قيمة تتعين لأي من عناصر التركيبة تكون هي القيمة التي تتعين لجميع باقي العناصر .

مثال (10.8.1)

ماذا ينتج عند تنفيذ البرنامج المبين بالشكل(10.8.1)؟

الإجابة: الذي ينتج هو طباعة الآتي :

```
2005  2005  2005
```

والمثال التالي يوضح طريقة استخدام هذا النوع من المتغيرات .

مثال (10.7.1) :

ماذا يطبع البرنامج التالي ؟

```
main()
{
        enum set { FALL, WINTER, SPRING, SUMMER} ;
        enum set season;
        for ( season =FALL; season<=SUMMER; season++)
                printf("\n %d ", season);
}
```

الشكل (10.7.1) برنامج النوع المسرود enum .

عند تنفيذ هذا البرنامج نحصل على الناتج التالي :

0
1
2
3

وقد نتوقع أن يطبع لنا عناصر الفئة set كاملة ، ولكن بدلاً من ذلك تمت طباعة الأعداد من 0 إلى 3 . وربما نعتقد أننا لو غيرنا %d في جملة الطباعة إلى %s سنحصل على ما نريد (أي سرد عناصر الفئة) ، ولكن ذلك غير صحيح وسيعطي نتائج خاطئة . والسبب في ذلك كله أنه عند تحديد فئة معينة

3 . عند تنفيذ البرنامج تحصل على الناتج التالي :

3 /4 + 4 /5 = 31 /20

وهي الإجابة الصحيحة .

10.7 النوع المسرود Enumeration type

النوع المسرود (ويسمى أيضا المعدود) هو الذي يمكن تحديد مجاله بفئة معدودة العناصر . فمثلاً إذا كان المتغير season يرمز لأحد فصول السنة :

{ fall , winter , spring , summer }

فإن هذه الفئة تحتوي على 4 عناصر فقط ، وبالتالي يعتبر season من النوع المعدود (المسرود) enumeration type .

و لتحديد فئة شاملة لمتغير ما نستخدم في لغة سي الكلمة المحجوزة enum متبوعة باسم الفئة ثم نضع عناصر الفئة مرتبة من اليسار إلى اليمين بين القوسين { } .

فمثلاً : إذا سمينا فئة الفصول الأربعة set، فإن الإعلان يكون على النحو التالي :

enum set { FALL , WINTER , SPRING , SUMMER } ;

والآن لإعلان أن season متغير مجاله الفئة set، نكتب :

enum set season ;

بعد هذا التعريف للنوع cmplx يمكننا تعريف متغيرات من هذا النوع على النحو التالي:

```
cmplx  z1 , z2,  w;
```

مثال(10.9.2) : في هذا المثال ، نقوم بكتابة برنامج لحساب الأعداد المركبة complex numbers . لاحظ أن العدد المركب يتكون من جزءين: حقيقي real وتخيلي imaginary.

وأن ناتج جمع أو طرح أو ضرب أو قسمة عددين مركبين هو أيضا عدد مركب.

يتكون البرنامج من الدوال التالية:

- الدالة menu تستقبل عددين مركبين من لوحة المفاتيح ، وترسلهما إلى الدالة main.
- الدالة add تقوم بجمع عددين مركبين.
- الدالة subtract تقوم بطرح عددين مركبين.
- الدالة multiply تقوم بضرب عددين مركبين.
- الدالة divide تقوم بقسمة عددين مركبين.

```
typedef struct
        { double real;
          double imag; } cmplx ;
```

للمقارنة مع التركيبة struct ، نقوم بتغيير كلمة union إلى struct ، ونعيد البرنامج ، لنحصل على الآتي :

2 9 2005

أي طباعة السنة ثم الشهر ثم اليوم وهو المطلوب .

من الواضح أن استخدام union بدلا من struct يوفر لنا مساحة من الذاكرة ، ولكن ذلك على حساب تعقيد البرنامج ، وإمكانية الوقوع في الخطأ .

10.9 تعريف النوع باستخدام typedef

إذا تم تعريف نوع بيانات (وليكن اسمه oldtype) باستخدام struct أو غير ذلك من طرق التعريف، فإن المؤثر typedef يمكّننا من تعريف نوع جديد (وليكن اسمه newtype) مكافئ للنوع oldtype وذلك على النحو التالي:

typedef oldtype newtype

نستفيد من هذه الخاصية في لغة سي بصورة خاصة عند تعريف السجلات بواسطة struct .

مثال(10.9.1) : يمكننا تعريف نوع جديد اسمه cmplx كالآتي:

```
typedef  struct { double real;  double imag;}      cmplx ;
```

حيث يرمز هذا النوع للأعداد المركبة من جزء حقيقي real وجزء تخيلي imag.

أي طباعة قيمة السنة 3 مرات بدلا من طباعة اليوم والشهر والسنة.

```
main()
{
        union date
        {
                int day;
                int month;
                int year;
        }today;
        today.day=2;
        today.month=9;
        today.year=2005;
        printf("\n %d %d %d ", today.day, today.month,
                                today.year);
}
```

الشكل (10.8.1) برنامج باستخدام union

وتفسير ذلك أن آخر قيمة تعينت هي 2005 للمتغير today.year ، وبذلك عند طباعة باقي العناصر (أي today.day , today.month) نتج عنها طباعة نفس قيمة today.year ، والسبب كما ذكرنا هو أن جميع العناصر للتركيبة union تشترك في نفس الموقع .

```
cmplx w;
double x1 = z1.real,
       y1 = z1.imag,
       x2 = z2.real,
       y2 = z2.imag;
w.real = x1*x2 - y1*y2;
w.imag = x1*y2 + x2*y1;
return(w);
}

cmplx divide ( cmplx z1 ,cmplx z2)
{
 cmplx w;
 double x1 = z1.real,
        x2 = z2.real,
        y1 = z1.imag,
        y2 = z2.imag;
 w.real = ( x1*x1 - y1*y2)/( x2*x2+y2*y2);
 w.imag = ( y1*x2 + y1*x1)/( x2*x2 +y2*y2);
 return(w);
}
```

الشكل (10.9.1) برنامج حساب الأعداد المركبة.


```
printf("\n first numb
scanf("%lf,%lf",&x1
printf("\n second nu
scanf("%lf,%lf", &x
z1->real = x1;
z1->imag = y1;
z2->real = x2;
z2->imag = y2;
printf("\n Please en
return;
}

cmplx add(cmplx z1
{ cmplx w;
 w.real = z1.real + z
 w.imag = z1.imag
 return(w);
}

cmplx subtract( cmp
{ cmplx w;
 w.real = z1.real - z
 w.imag = z1.imag
 return(w);
}


cmplx multiply( cm
{
```

```
  printf("\n first number : real part, imag part-->");
  scanf("%lf,%lf",&x1, &y1);
  printf("\n second number : real part, imag  part-->");
  scanf("%lf,%lf", &x2, &y2);
  z1->real = x1;
  z1->imag = y1;
  z2->real = x2;
  z2->imag = y2;
  printf("\n Please enter operation:  +  -  *   / ") ;
  return;
}

cmplx add(cmplx z1 , cmplx z2)
{  cmplx w;
  w.real = z1.real + z2.real;
  w.imag = z1.imag + z2.imag;
  return(w);
}

cmplx subtract( cmplx z1 , cmplx z2)
{ cmplx w;
 w.real = z1.real - z2.real;
 w.imag = z1.imag  - z2.imag;
 return(w);
}


cmplx multiply( cmplx z1 , cmplx z2 )
{
```

```
main()
{  cmplx  z1,z2, w;
char choice;
void menu( cmplx *z1, cmplx *z2);
cmplx add( cmplx , cmplx),
          subtract( cmplx , cmplx),
          multiply ( cmplx , cmplx),
          divide( cmplx , cmplx);
menu( &z1, &z2 );

choice = getche();
switch( choice )
{
   case '+' :  w = add(z1,z2); break;
   case '-' :  w = subtract(z1,z2);  break;
   case '*' :  w = multiply(z1,z2);  break;
   case '/' :  w = divide(z1,z2);  break;
   default :
        printf( " \choice not valid "); exit();
  }
  printf( "\n result :  %f   %f    " ,w.real, w.imag);
   getch();

}

void  menu(cmplx *z1, cmplx *z2)
{ double x1, x2 , y1, y2;
  printf("\n this program for complex arithmetic");
  printf("\n Please enter 2 complex numbers:");
```

استخدم التركيبة cmplx التي تحتوي على عضوين من النوع float هما الجزء الحقيقي والجزء التخيلي للجذر ، بحيث تكون الدالة roots من النوع cmplx

ملاحظة :

إذا كان الجذران حقيقيين فإن الجزء التخيلي يساوي صفراً .

9 . اكتب الدالة add_cmplx التي تقوم بجمع عددين مركبين يتكون كل منهما من جزء حقيقي وجزء تخيلي ، علما بأن ناتج جمع عددين مركبين هو أيضاً عدد مركب ويساوى جزؤه الحقيقي مجموع الجزءين الحقيقيين ، وجزؤه التخيلي مجموع الجزءين التخيليين .

10 . اكتب الدالة mul_cmplx التي تقوم بضرب عددين مركبين Z1 و Z2 على التوالي .

11 . أعد كتابة تمرين (9) بحيث تكون الدالة add_cmplx من النوع الفارغ void، يتم ترجيع حاصل الجمع في بارامتر Z3 وذلك باستخدام الاستدعاء بالعنوان .

12 . اكتب الدالة age التي تقوم بحساب العمر بالأيام والأشهر والسنوات وذلك

5 . اكتب برنامجا لقراءة عنوان هذا الكتاب واسم المؤلف وطباعة الأتي :

Book Title : .....

Book Author : .....

حيث يطبع في الفراغ الأول عنوان الكتاب وفي الفراغ الثاني اسم المؤلف بالكامل . استخدم تركيبة اسمها book للكتاب ، وتركيبة اسمها author للمؤلف .

6 . لديك قائمة بأرقام حسابات 10 زبائن بالمصرف ورصيد كل منهم ، والمطلوب حساب متوسط الرصيد مستخدما تركيبة customer التي تتكون من الرقم والرصيد ، ومصفوفة balance من النوع customer لتخزين بيانات الزبائن في جدول بحث .

7 . اكتب الدالة day_month_year من النوع date الذي يتكون من 3 أعضاء : (day , month , year) أي اليوم والشهر والسنة ، وذلك لحساب عدد الأيام والأشهر والسنوات في عدد معلوم من الأيام . لتبسيط الحساب افترض أن الشهر = 30 يوما . استخدم هذه الدالة لحساب عدد السنوات والأشهر والأيام في 30,000 يوم .

8 . اكتب الدالة root التي تحسب جذري المعادلة

$$ax^2 + bx + c = 0$$

10.10 تمارين

1 . ما الفرق بين المصفوفة array والتركيبة structure ؟

2 . اكتب تركيبة مناسبة بلغة سي لتعريف الآتي :

أ . الزمن (ساعة ، دقيقة ، ثانية) .

ب . المسكن (العنوان ، عدد الحجرات ، المساحة ، الثمن) .

جـ . الكتاب (اسم الناشر ، اسم المؤلف ، العنوان ، الثمن ، سنة النشر) .

3 . اكتب برنامجا لحساب عدد الساعات والدقائق والثواني في فترة زمنية معلومة بالثواني . استخدم تركيبة مناسبة لمتغير الزمن .

4 . اكتب الدالة timer التي تحسب الفرق بالساعات والدقائق والثواني بين الزمن t1 والزمن t2 ، حيث t1 و t2 هما معطيات الدالة ، واستخدمها لحساب الفرق بين الساعة 1 و 55 دقيقة و 34 ثانية ، والساعة 5 و 24 دقيقة و 47 ثانية .

بمعلومية تاريخ الميلاد birth ، وتاريخ اليوم today ، وهما من النوع date الذي يتم تعريفه في البرنامج متكونا من اليـــوم والـــشهر و الـــسنة .استخدم طريقتين :

أ . افترض أن الدالة age من النوع date .

ب . افترض أن الدالة age من النوع void .

13 . إذا كان المتغير month من النوع المعدود ومجاله الفئة :

months = { Jan , Feb , Mar , Apr , May , Jun , Jul , Aug , Sep , Oct , Nov , Dec }

اكتب برنامجا يستخدم هذا المتغير لطباعة كل شهر من أشـــهر الـــسنة والفصل الذي يقع فيه . مثلاً :

Jan is in winter

Feb is in winter

Mar is in spring

.....

وهكذا .....

14 . إذا كانت بيانات قمصان من نوع shirts تتكون من الآتي :

أ . الحجم (صغير S ومتوسط M وكبير L)

ب . اللون (أبيض WHITE ، أسود BLACK ، أزرق BLUE ، أصفر YELLOW ، أخضر GREEN)

ج . الثمن (من النوع الكسري float)

استخدم جدول البحث لإدخال بيانات عن 15 عينة ، ثم أوجد :

أ . عدد القمصان من الحجم 5 .

ب . عدد القمصان ذات الحجم M واللون BLUE .

ج . متوسط أثمان جميع القمصان .

15 . اكتب معاني المصطلحات التالية :

structure

record

look_up table

enumeration type

union