Structs, “strukturer”

- Antag att du skall skriva ett program som hanterar **ett** datum (dag, månad, år)
- Hur skulle du representera detta datum i ditt program?
- Antag att ditt program skall hantera **100** datum. Hur skulle du representera dessa?

- Istället för att använda separata variabler för olika delar av datumet kan en **`struct`** användas för att skapa en *sammansatt datatyp*

En struktur för datum

- I stället för att definiera tre olika variabler kan vi definiera en ny *datatyp* som *innehåller* uppgifter om dag, månad och år
- Därefter kan vi deklarera *en* variabel som innehåller alla dessa uppgifter
- "Delarna" som ingår i denna nya datatyp benämns "members" (medlemmar)

```
/* Declare a new struct
   and name it "date_t" */
struct date_t
{
   int day; /* Member #1 */
   int month; /* Member #2 */
   int year; /* Member #3 */
}; /* Note the semicolon! */

/* Declare a variable of
   type "struct date_t" */
struct date_t date;
```

En struktur för datum

- För att komma åt de olika delarna av datumet används en speciell syntax:
  variabelns namn,
  *följt av en **punkt***,
  följt av delens namn

```
/* Declare a variable of
   type "struct date" */
struct date_t date;

/* Initialize the variable */
date.day = 24;
date.month = 10;
date.year = 2017;

/* Output the values */
printf("Date is %d/%d/%d\n",
date.day,
date.month,
date.year);
```

Initialisering av structs

```
/* A struct can also be initialized in a
   similar fashion as an array: */
struct date_t date = { 24, 10, 2017 };

/* ...Or like this: */
struct date_t date = { .day = 24,
                       .month = 10,
                       .year = 2017 };

/* The former syntax also enables you to
   initalize only some members, and in any order: */
struct date_t date = { .month = 10, .day = 24 };
```

Användning av structvariabler

- Medlemsvariablerna i en struct kan läsas och uppdateras exakt som vanliga variabler:

```
/* Increase date by one, assuming we have the
   daysInMonth lookup table available.
   Note: the example below is not complete! */

if (daysInMonth[date.month] == date.day)
{
   date.month++;
   date.day = 1;
}
else
{
   date.day++;
}
```

Strukturer och funktioner

- **`struct`**-variabler kan förstås också skickas som argument till funktioner
- För att använda en struct i flera funktioner måste deklarationen av datastrukturen göras **globalt**, på samma sätt som för globala variabler
- Med hjälp av en **`struct`** kan man returnera flera enskilda värden från en funktion
  - (Ett annat alternativ för att åstadkomma detta är med hjälp av *pekare*, som behandlas senare i kursen)

Strukturer och funktioner

```
int calcDateDifference(struct date_t d1,
                       struct date_t d2)
{
   int difference = 0;
   // Actual calculation of 'd1 – d2' omitted...
   return difference;
}
```

```
struct date_t firstDate, secondDate;
// Initialization of dates omitted...

int result = calcDateDifference(firstDate, secondDate);
printf("Difference between the dates is %d days\n",
                                              result);
```

Strukturer och funktioner

```
struct date_t calcDateDifference( struct date_t d1,
                                  struct date_t d2 )
{
   struct date_t difference = 0;
   // Actual calculation of 'd1 – d2' omitted...
   return difference;
}
```

```
struct date_t firstDate, secondDate;
// Initialization of dates omitted...

struct date_t result = calcDateDifference( firstDate,
                                           secondDate );
printf("Difference between the dates is \\
        %d year(s), %d month(s) and %d day(s)\n",
        result.year, result.month, result.day);
```

Strukturer och räckor

- Strukturer kan placeras i räckor –
  en räcka kan endast innehålla en viss strukturtyp:

```
// Declare an array of 100 'struct date_t' elements
struct date_t dateArray[100];

// Initialize the 1st element – must use a type cast
dateArray[0] = (struct date_t){24, 10, 2017};

// Set the year member of the 2nd element
dateArray[1].year = 2017;

// Loop through the array, output day members
for (i=0; i<100; i++)
{
   printf("%d", someDates[i].day);
}
```

Strukturer som innehåller strukturer

- En struktur kan innehålla en annan struktur:

```
struct time_t {
   int hour;
   int minute;
   int second;
};
struct timestamp_t {
   struct date_t date;
   struct time_t time;
};
// Initialize both date and time 'substructs'
struct timestamp_t ts = { {24,10,2017}, {12,30,00} };
// Change the hour value
// Note the 'double-dot' syntax
ts.time.hour = 13;
```

Typedef

- Kan använda **`typedef`** för att ge ett 'alias' åt en viss typ
- Användbart i samband med strukter – behöver inte ange **`struct`** vid deklaration av variabler

```
typedef struct {
    int hour;
    int minute;
    int second;
} time_t;

typedef struct {
    int day;
    int month;
    int year;
} date_t;
```

```
typedef struct  {
    time_t time;
    date_t date;
} timestamp_t;

time_t t = { 12, 30, 0 };
t.hour = 13;

timestamp_t ts = ...
```