Föreläsningsanteckning 6
DA105A - Programmering med C, grundkurs, 7.5 hp

..

Innehåll

1 Inledning

Denna föreläsning behandlar fält. Föreläsningen ger information om hur fält kan deklareras och användas, samt hur fält kan användas som parametrar till funktioner. Föreläsningen behandlar material som ingår i kapitel 6 i kursboken *C How to Program - Fifth Edition* av H.M. Deitel och P.J. Deitel.

Beteckningen [DEITEL] används nedan för att referera till denna bok.

Föreläsningen använder bostadskalkyl-programmet som beskrivits i tidigare föreläsningar, och exemplifierar hur fält kan användas för att modifiera detta program. Modifieringarna utgår från den version av programmet som behandlats i föreläsning 5 [1].

2 Fält

Ett fält är en datastruktur som innehåller ett antal element, där alla element är av samma datatyp. Ett fält kallas även för *vektor* eller *array*, som också är den engelska beteckningen. Man kan använda fält i programmet för bostadskalkyl-beräkningar, t.ex. för att lagra den månatliga räntan för en följd av år, eller för att lagra den återstående skulden för en följd av år.

Ett fält har en specificerad storlek. I programspråket C kan man definiera fält med en på förhand definierad storlek. Detta kommer att göras i denna föreläsning. Man kan också definiera fält vars storlek kan variera under den tid som programmet exekverar. Detta kan göras genom att använda *dynamisk minnesallokering*, som behandlas senare i kursen.

Elementen i ett fält kan läsas och skrivas genom att använda ett *index*. Ett sådant index är ett *heltal*, som antar värden mellan 0 och *fältets storlek*

minus ett. Man använder ett index för att peka ut ett enskilt element i ett fält. Indexvärdet noll används för att peka ut fältets *första* element, och ett indexvärde som är fältets storlek minus ett används för att peka ut fältets *sista* element.

3 Deklaration av fält

Ett fält kan definieras i programspråket C, t.ex. enligt

```
/* interest cost per month */
double monthly_interest_cost[MAX_N_YEARS];
```

Ovanstående deklaration definierar ett fält med element av typen *double*. Fältets namn är *monthly_interest_cost*, vilket indikerar att fältet kan användas för att lagra den månatliga räntekostnaden. Fältets *storlek* anges med en symbol *MAX_N_YEARS*. Denna är definierad med ett *#define*-direktiv, enligt

```
/* maximum number of years for cost calculations */
#define MAX_N_YEARS 100
```

Detta innebär att fältet *monthly_interest_cost* blir en datastruktur som kan lagra 100 stycken flyttal, av typen *double*. Fältet kan därmed lagra den månatliga räntekostnaden för maximalt 100 år.

4 Användning av fält

Man kan använda ett fält i programkod, t.ex. för att läsa element eller för att skriva värden till element. Man skulle t.ex. kunna använda sig av en *for*-sats för att initialisera fältet *monthly_interest_cost*, som deklareras enligt ovanstående beskrivning. En sådan initialisering används för att säkerställa att alla fältets element har tilldelats värden. En initialisering som ger alla element värdet noll kan göras enligt

```
/* initialise monthly interest cost */
for (i = 0; i < MAX_N_YEARS; i++)
{
    monthly_interest_cost[i] = 0;
}
```

där en räknare, kallad *i*, används för att indexera fältets element. Man kan notera att denna räknare antar värden från 0 till och med *MAX_N_YEARS-1*.

Man kan läsa om fält, hur de kan definieras och användas, i avsnitten 6.1 till 6.4 i [DEITEL].

5 Användning av fält som parametrar till funktioner

Ett fält kan användas som parameter till en funktion. Ett enkelt exempel på en funktion som har ett fält som parameter kan vara en funktion som utför en initialisering av ett fält, genom att tilldela ett specificerat värde till fältets alla element. En sådan funktion kan skrivas enligt

```
/* fill_double_array: fills the array arr, having the
   length length, with the value value */
void fill_double_array(double arr[], int length, double value)
{
    /* loop counter */
    int i;

    for (i = 0; i < length; i++)
    {
        arr[i] = value;
    }
}
```

Denna funktion kan sedan anropas, t.ex. för att initialisera fältet *monthly_interest_cost* med värdet noll, enligt

```
    /* initialise monthly interest cost */
    fill_double_array(monthly_interest_cost, MAX_N_YEARS, 0);
```

Ett annat exempel på en funktion, som använder flera fält som parametrar, kan vara en funktion som beräknar olika typer av kostnader för ett antal år framåt i tiden. En sådan funktion, kallad *calculate_future_data*, som beräknar den månatliga räntekostnaden, den återstående mängden pengar per månad, samt resterande skuld, för ett antal år framåt i tiden, kan ha parametrar enligt

```
/* calculate_future_data: calculates monthly interest cost,
   monthly balance and debt for this year and the following
   n_years years, and stores the calculated data in arrays
   monthly_interest_cost, monthly_balance, and net_debt */
void  calculate_future_data(
    /* input arguments related to money */
    double price, double interest_p, double monthly_income,
    double monthly_instalment, double monthly_rent,
    /* number of years */
    int n_years,
    /* output arguments */
    /* the monthly interest cost */
```

```
double monthly_interest_cost[],
/* the remains, for food, bills and pleasure */
double monthly_balance[],
/* the remaining debt */
double net_debt[])
```

Fält som används som parametrar i programspråket C överförs via *referensanrop*. Detta innebär att det fält som används som argument för en parameter, i själva anropet, *inte* kopieras till den funktion som anropas. Istället överförs fältets adress. Denna funktionalitet gör det möjligt att *ändra* innehållet i ett fält som används som argument. Detta är ofta en önskvärd effekt, t.ex. i funktionerna *fill_double_array* och *calculate_future_data*, som beskrivs ovan, eftersom dessa funktioner skall förändra värden som finns lagrade i fält som anges som parametrar.

Andra typer av parametrar i programspråket C, t.ex. heltal, flyttal och tecken, överförs via *värdeanrop*, som innebär att värdet på ett argument kopieras i samband med ett funktionsanrop. Sådana parametrar kan därmed inte ändras inifrån en funktion. Om detta önskas får man istället använda *adressen* till en variabel som parameter, för att åstadkomma ett referensanrop. Detta har beskrivits kort i avsnitt 3.2 i föreläsning 5 [1], och behandlas i avsnitt 7.4 i [DEITEL].

Användning av fält som parametrar till funktioner beskrivs i avsnitt 6.5 i [DEITEL].

Referenser

[1] Föreläsningsanteckning 5