

4) Sammansatta datatyper

float x[3] -- x[0], x[1], x[2] -- 3 reella tal
int nr[4] -- nr[0], nr[1], nr[2], nr[3] -- 4 heltal
char ch[5] -- ch[0], ch[1]...ch[4] -- 5 tecken

Vektor (fält, array, indexerad variabel)

Ex) Program som läser 3 reella tal till en vektor och skriver talen i omvänd ordning.

```
void main()  
{
```

```
float x[3];  
int i;
```

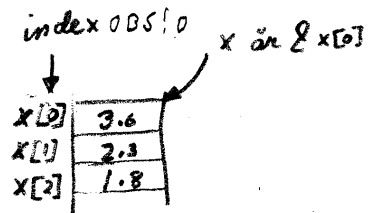
```
for(i=0; i<3; i++)
```

```
{ printf("Ge x[%d]: ", i); // Ge x[0]: 3.6  
scanf("%f", &x[i]); // Ge x[1]: 2.3  
// Ge x[2]: 1.8  
}
```

```
for(i=2; i>=0; i--)
```

```
{ printf("%0.1f\n", x[i]);  
}
```

```
getch();  
}
```



ⓔ Program som läser 7 reella tal till en vektor och som beräknar medelvärdet av de tal som återstår då man tagit bort det minsta och det största talet.

```
void main()
{
    float x[7], sum, min, max;
    int i;

    printf("Ge 7 reella tal: ");
    for(i=0; i<7; i++)
    {
        scanf("%f", &x[i]);
    }

    sum = min = max = x[0];
    for(i=1; i<7; i++)
    {
        sum += x[i];
        if (x[i] < min)
            min = x[i];
        else if (x[i] > max)
            max = x[i];
    }
    printf("Medel = %.2f", (sum - min - max) / 5.0);
    getch();
}
```

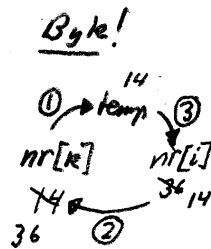
②

Ex) Initiera en heltalsvektor och sortera vektorn i stigande ordning

```

void main()
{
    int i, k, temp, nr[5] = {36, 14, 28, 15, 26};
    for(i=0; i<4; i++)
    {
        for(k=i+1; k<5; k++)
        {
            if(nr[k] < nr[i])
            {
                temp = nr[k];
                nr[k] = nr[i];
                nr[i] = temp;
            }
        }
    }
}

```



Variabel värden

- nr[0] = 36 (14)
- nr[1] = 14 36 28 (15)
- nr[2] = 28 28 36 36 28 (26)
- nr[3] = 15 15 15 28 36 36 (28)
- nr[4] = 26 26 26 26 26 28 (36)

Sträng

(Ex) a) En vektor av tecken

```
char ch[3] = {'H', 'e', 'j'};
```

b) Motvarande sträng

```
char str[4] = {'H', 'e', 'j', '\0'}; ASCII 0
```

eller

```
char str[4] = "Hej"; // \0 sätts automatiskt
```

(Ex) Läs en sträng och skriv den baklänges

```
void main()
```

```
{
```

```
    char str[10];
```

```
    int i, k;
```

```
    printf("Ge en sträng: ");
```

```
    scanf("%s", str); // OBS! Inget &
```

```
    i = 0;
```

```
    while (str[i] != '\0')
```

```
    {
```

```
        i++;
```

```
    }
```

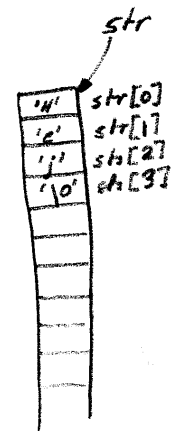
```
    for (k = i - 1; k >= 0; k--)
```

```
    {
```

```
        printf("%c", str[k]);
```

```
    }
```

```
}
```



Ⓔ Läs en textrad och räkna antalet mellanslag på raden.

Körex

Ge en textrad: Hej på dig!

Antal mellanslag = 2

Program

```
void main()
{
    char text[80];
    int i, antal = 0;
    printf("Ge en textrad: ");
    gets(text); // Läser till radslut
                // scanf läser till mellanslag
    for(i=0; text[i] != '\0'; i++)
    {
        if (text[i] == ' ')
        {
            antal++;
        }
    }
    printf("Antal mellanslag = %d", antal);
    getch();
}
```

⑤

Struct (post, record)

ⓔx) Avbilda en motor med antal cylindrar och effekt. Läs och skriv data.

```
void main()
```

```
{  
  struct  
  {  
    int cyl;  
    float effekt;  
  } m;
```

Alternativ!

```
struct motor  
{  
  int cyl;  
  float effekt;  
};  
struct motor m;
```

```
printf("Ge antal cylindrar: ");
```

```
scanf("%d", &m.cyl);
```

```
printf("Ge effekt: ");
```

```
scanf("%f", &m.effekt);
```

m	
4	m.cyl
120	m.effekt

```
printf("Antal cylindrar = %d\n", m.cyl);
```

```
printf("Motoreffekt = %.1f", m.effekt);
```

```
getch();
```

```
}
```

⑥

Ⓔ Initiera en vektor av fem motorer och sortera den i första hand efter antal cylindrar och om lika antal cylindrar efter effekt.

```
void main()
{
    struct motor
    {
        int cyl;
        float effekt;
    };
    struct motor m[5] = { {4, 120}, {4, 100},
                        {6, 150}, {6, 140},
                        {4, 110} }, temp;

    int i, k;

    for (i=0; i<4; i++)
    {
        for (k=i+1; k<5; k++)
        {
            if (m[k].cyl < m[i].cyl)
            {
```

```

temp = m[k];
m[k] = m[i];
m[i] = temp;
}
else if (m[k].cyl == m[i].cyl &&
         m[k].effekt < m[i].effekt)
{
temp = m[k];
m[k] = m[i];
m[i] = temp;
}
}
}
for (i=0; i<5; i++)
{
printf("Cylindrar = %d\n", m[i].cyl);
printf("Effekt = %.1f\n\n", m[i].effekt);
}
getch();
}

```

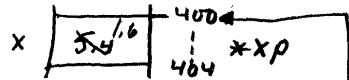
OBS! Motorena skrivs ut i stigande ordning
vad gäller cylindrar och för varje cylindrantal
i stigande effekt.

Pekare (adressvariabel)

Ex a) Reellt tal

float x;

x = 5.4;



b) Pekare till reellt tal

float *xp; // Datatyp float *

xp = &x; // Tilldela adress
// ex 400



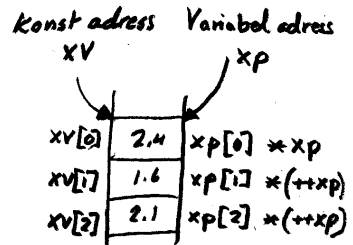
*xp = 1.6; // Avrefererad pekare
// detsamma som x

c) Reell vektor

float xv[3] = {2.4, 1.6, 2.1};

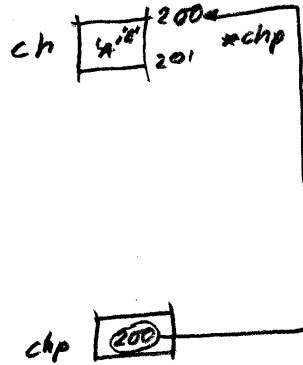
float *xp;

xp = xv;



Ex a) Tecken

```
char ch;  
ch = 'A';
```

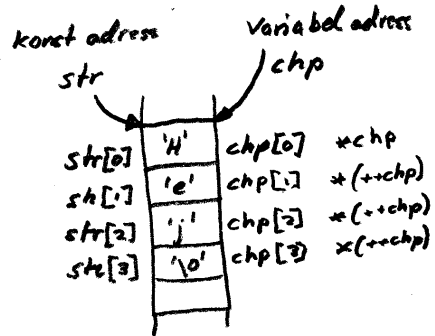


b) Pekare till tecken

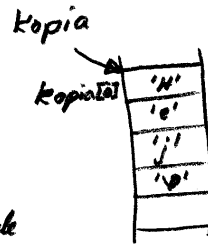
```
char *chp;  
chp = &ch;  
*chp = 'B';
```

c) stäng

```
char str[5] = "Hej";  
char *chp;  
chp = str;  
char kopia[5];
```

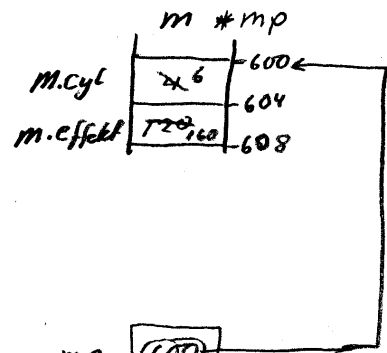


~~kopia = str;~~ // Fel. Kopia är
// en konstant
// adress och kan ej
// tilldelas nytt värde
// Kopiera tecken för tecken



(Ex) a) struct

```
struct motor
{
    int cyl;
    float effekt;
};
struct motor m = {4, 120};
```



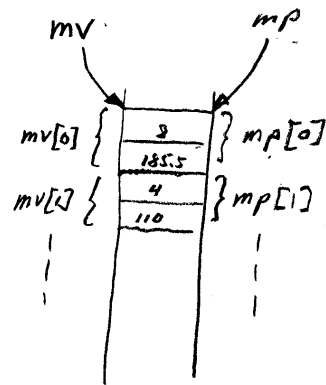
b) Pekare till struct

```
struct motor *mp;
mp = &m;
(*mp).cyl = 6; // Alt mp -> cyl = 6;
(*mp).effekt = 160; // Alt mp -> effekt = 160;
```



c) Vektor av structar

```
struct motor mv[5];
mv[0].cyl = 8;
mv[0].effekt = 185.5;
...
struct motor *mp;
mp = mv;
mp[0].cyl = 4;
mp[1].effekt = 110;
```



Hemuppgift

Program som simulerar en lottodragning med 35 bollar numrerade från 1-35 som släpps av "Dracula" och bildar en lottorad med 7 nummer

```
void main()
```

```
{  
    int lottobollar[36], antal_slappade=0,  
        i, varu, temp, k;
```

```
    /* skapa vektorn lottobollar med  
       samma elementvärden som  
       index alltså 1 till 35 */
```

```
    /* Blanda lottobollarna genom  
       att slumpa två indexvärden  
       mellan 1 och 35 och sedan  
       byta elementvärden mellan dem.  
       Gör detta 1000 gånger */
```

```
    /* Släpp boll nr 35 och skriv  
       dess elementvärde. Minska antal  
       bollar till 34, blanda och släpp  
       osv */
```

```
}
```

(12)