

Datorsystemteknik

för D2, ICT2, E3 och Mek3

Nicholas Wickström

Högskolan i Halmstad
Sverige

Outline

- Sammanfattning av föregående fl.
- Pekare
- Vektorer
- Pekararitmetik
- Funktioner och pekare
- Pekare till funktioner

Def. Pekare

En pekare är en variabel som innehåller adressen till en variabel.

Pekare, exempel

```
int nVar1 = 1, nVar2 = 99;
int *pnTemp; /* int pekare */

pnTemp = &nVar1;
/* pnTemp pekar på nVar1 */
/* &nVar1 är adressen till nVar1 */

nVar2 = *pnTemp;
/* Tilldelning av värdet på nVar1 till nVar2 */
/* via pekaren pnTemp (nVar2 = 1) */
```

Pekare, exempel (forts.)

```
int nVar1 = 1, nVar2 = 99, nVect[10];
int *pnTemp; /* int pekare */

pnTemp = &nVar1; /* pnTemp pekar på nVar1      */
                /* &nVar1 är adressen till    */
                /* nVar1                       */

*pnTemp = 0;     /* nVar1 är nu 0              */

pnTemp = &nVect[0]; /* pnTemp pekar nu på nVect[0], */
                /* första elementet i vektorn */
```

Pekare och operatörer

```
int nVar1 = 1, nVar2 = 0, *pnTemp;
```

```
pnTemp = &nVar1;
```

```
*pnTemp = *pnTemp + 19;
```

```
/* På samma sätt som nVar1 = nVar1 + 19 */
```

```
nVar2 = *pnTemp + 1;
```

```
/* På samma sätt som nVar2 = nVar1 + 1 */
```

```
*pnTemp += 1; ++*pnTemp; (*pnTemp)++; /* Samma sak */
```

```
/* Varning!! *pnTemp++ och (*pnTemp)++ betyder */
```

```
/* olika saker!! (Pekararitmetik) */
```

Vektorer

```
int nVect[10], *pnTemp, nVar;
```

```
pnTemp = &nVect[0];
```

```
/* pnTemp pekar nu på nVect[0],      */
```

```
/* första elementet i vektorn      */
```

```
nVar = *pnTemp;
```

```
/* tilldeln. av nVect[0] till nVar  */
```

```
nVar = *(pnTemp+1);
```

```
/* tilldeln. av nVect[1] till nVar  */
```

```
/* Notera! pnTemp pekar fortfarande på nVect[0] */
```

Pekararitmetik

```
int nVect[10], *pnTemp, nVar;

pnTemp = &nVect[0]; /* pnTemp pekar nu på nVect[0], */
                  /* första elementet i vektorn */

nVar = *pnTemp;
/* tilldeln. av nVect[0] till nVar */

nVar = *pnTemp++;
/* tilldeln. av nVect[1] till nVar */

/* Notera!! Nu har pekaren flyttats till nVect[1] */
```


Pekare till pekare

```
int nMatrix1[5][10];
```

```
int *pnMatrix2[5];
```

```
/* nMatrix1 upptar 5*10 element i minnet */
```

```
/* för pnMatrix2 allokeras bara plats för 5 pekare */
```

```
/* I nMatrix1 fallet måste varje rad vara 10 element */
```

```
/* Medan i fallet pnMatrix2 kan raderna variera */
```

Dynamisk minnesallokering

```
#include <stdlib.h> /* för malloc, calloc samt free */

int *pnVector;

pnVector = (int *)malloc(sizeof(int) * nNumElements);
if (pnVector == NULL)
/* Kan inte allokera minne, panik! */
...

free(pnVector); /* Glöm ej avallokering av minnet */
}
```

Pekare i funktionsanrop

```
void CalculateSum(int *nArg1, float *fArg2, float *fRes)
{
    *fRes = (float)(*nArg1) + *fArg2;
}
```

```
/* Anropas enligt: */
```

```
CalculateSum(&nTal1, &fTal2, &fMyResult);
```

Pekare till funktioner

```
extern void add(void);  
extern void mult(void);  
extern void sub(void);
```

```
void (*Calc[3])() = {add, mult, sub};
```

```
Calc[0](); /* anrop till funktionen add */
```

Pekare till funktioner (forts.)

```
extern int add(int nArg1, int nArg2);  
extern int mult(int nArg1, int nArg2);  
extern int sub(int nArg1, int nArg2);  
int nRes;  
  
int (*Calc[3])(int nArg1, int nArg2) = {add, mult, sub};  
  
nRes=Calc[0](1,2); /* anrop till funktionen add */
```