Computer systems engineering I

Nicholas Wickström

IDE, Halmstad University

2009

Outline

Def. Pointer

A pointer is a variable containing the adress to another variable, function or adress.

Pointer, exempel

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

pnTemp = &nVar1;
/* pnTemp points onnVar1                          */
/* &nVar1 is the address of nVar1                 */

nVar2 = *pnTemp;
/* Assignment of the value of nVar1 to nVar2 */
/* via the pointer pnTemp (nVar2 = 1)        */
```

Pointer, exempel (forts.)

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

pnTemp = &nVar1; /* pnTemp points on nVar1    */
                 /* &nVar1 is the address to  */
                 /* nVar1                     */

*pnTemp = 0;     /* nVar1 is set to 0         */

pnTemp = &nVect[0]; /* pnTemp points on nVect[0],  */
                    /* the first element of vector */
```

Pointers andf operators

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

pnTemp = &nVar1;
*pnTemp = *pnTemp + 19;
/* similar to nVar1 = nVar1 + 19 */

nVar2 = *pnTemp + 1;
/* similar to nVar2 = nVar1 + 1 */

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

/* Warning!! *pnTemp++ and (*pnTemp)++ is not the  */
/* same!! (pointer arithmetics)                    */
```

Vektorer

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

pnTemp = &nVect[0];
/* pnTemp points on nVect[0],       */
/* the first element in the vector  */

nVar = *pnTemp;
/* assigment of nVect[0] to   nVar  */

nVar = *(pnTemp+1);
/* assignment of nVect[1] to  nVar  */

/* Note! pnTemp still at nVect[0]   */
```

Pointer arithmetics

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

pnTemp = &nVect[0]; /* pnTemp point on nVect[0],    */
                    /* first element in thevector   */

nVar = *pnTemp;
/* assignment of nVect[0] to nVar   */

nVar = *pnTemp++;
/* assignment of nVect[1] to nVar   */

/* Note!! Now the point moved to nVect[1] */
```

Pointers to pointers

```
int nMatrix1[5][10];
int *pnMatrix2[5];

/* nMatrix1 5*10 elements in memory                    */
/* pnMatrix2 5 pointers allocated in memory            */

/* In nMatrix1 ech row is 10 elements                  */
/* In pnMatrix2 they can be any length                 */
```

Dynamic memory

```
#include <stdlib.h> /* for malloc, calloc and free */

int *pnVector;

pnVector = (int *)malloc(sizeof(int) * nNumElements);
if (pnVector == NULL)
/* ERROR: can not allocate requested memory! */
...

free(pnVector); /* must return claimed memory */
}
```

Pointers in function calls

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

/* called by: */
CalculateSum(&nVal1, &fVal2, &fMyResult);
```

Vectors of pointers to function

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

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

Calc[0](); /* call function add */
```

Vectors of pointers to function

```
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); /* call function add with param */
```