Computer systems engineering I

Nicholas Wickström

IDE, Halmstad University

2009

Outline

if, if-(then)-else

```
/* if sats */
if (n > 0) {
   fAvg = fSum / (float)n;
   }

/* if-then-else */
if (n >= 0)
   nPositive = TRUE;
else
   nPositive = FALSE;
```

if, if-(then)-else, cont.

```
/* nested if-then-else sats, multiple choice */

if (nColor == YELLOW)
   nYellow++;
else if (nColor == BLUE)
   nBlue++;
else if (nColor == GREEN)
   nGreen++;
else
   nOthers+;
```

switch-case

```
switch (nColor) { /* switch-sats, multiple choice */
   case YELLOW:
      nYellow++;
      break;
   case BLUE:
      nBlue++;
      break;
...
   default:
      nOthers+;
      break;
}
```

while

```
/*  (n!) */
nfak = i = 1;
while (++i <= n)
   nfak *= i;
```

do-while

```
do {
   if (x-y < 0)
      nError += nSmallAdjust;
   else
      nError -= nSmallAdjust;
} while (nError > MAX_ERR ||
         nError < -MAX_ERR)
```

for

```
for (expr1; expr2; expr3)
   sats

equivalent too

expr1;
while (expr2) {
   sats
   expr3;
}
```

for cont.

```
/* x raised to n */
for (i = 1; i <= n; i++)
   nResult *= x;

/* (n!) (for-version) */
for (i = 1; i <= n; i++)
   nfak *= i;
```

Operators

```
nAntal++; /* Ekv. nAntal = nAntal + 1; */
++nAntal; /* Ekv. nAntal = nAntal + 1; */

i = 4; j = 7;
a = ++i * --j; /* a => 30 */
i = 4; j = 7;
a = i++ * j--; /* a => 28 */
```

Operators (comparison)

```
<   >   <=   >=   ==   !=

x < y   a > 3.5   i + j <= 4
x == y   i != 0
```

Operators (Logic)

```
&&   ||   !

AND, OR, NOT

x > 0 || !(i == 2 && j < 4)

NOTE!! i == !!i;
/* not allways true */
```

Operators (Bit)

```
~   <<   >>   &   ^   |

BNEG, LSHFT, RSHFT, BAND, BXOR, BOR
c = 0x8F;  /* 10001111 */
c2 = ~c    /* 01110000, or 0x70 */

c3 = c >> 2 /* 00100011 */

k = k | 0x20; /* Set 6e biten  */
k = k & 0xFFDF; /* Reset 6e biten */
```

Operators (Assignment)

```
i = 0; /* Simple assignment */
j = i; /* Alt. i = j = 0; */

i = i + j; /* Alt. i += j; */
```

Arrays, matrices

```
int nVector[100];
int nMatrix[3][3] =
{ { 1, 2, 3}, {4, 5, 6}, {7, 8, 9} };
int nTranspose[3][3];
/* avg of vector */
for (i = 0; i < 100; i++)
   nSum += nVector[i]/100.0;

/* Transpose of nMatrix */
for (i = 0; i < 3; i++)
   for (j = 0; j < 3; j++)
      nTranspose[i][j]=nMatrix[j][i];
```

Functions

```
float calc_avg (float x, float y)
{
   return ((x + y) / 2.0);
}

/* function call  */
nAvg = calc_avg(a, b);
/* limitation! only possible to return on arg.*/
/* use a struct */
/* best solved with pointers! */
```

Pre processor

```
#define N 1000
/* Macro */

#define TRUE 1
#define FALSE !TRUE

#ifdef DEBUG
/* print debug info */
#endif
```

Preprocessor cont.

```
#ifndef _MYHEADER_H_
#define _MYHEADER_H_

/* include file here */

#endif

/* Preprocessorn, before compilation */
```