C-ohjelmointi, kevät 2006

Kertausta

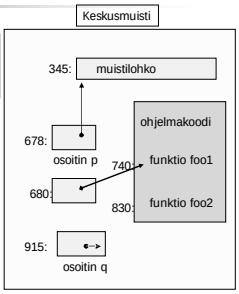
Luento 10
13.10.2006

Lyhyt kertaus osoittimista

```
char *p;  /* char, int, jne ilmoittavat, minkä tyyppisiä */
int *q;   /* olioita sisältäviin muistilohkoihin osoittavat */
Nämä määrittelyt varaavat tilan vain osoittimelle!
```

```
char *p = "Tätä varten varataan muistitilaa";
int luvut[] = {1, 2, 3, 4, 5};
double taulu[100];
Nämä varaavat tilaa myös muistilohkolle sekä
asettavat osoittimen osoittamaan ko. muistilohkoa.
```

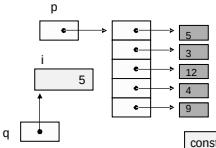
Tilaa voidaan varata myös malloc- ja calloc-funktioilla ja samalla asettaa jokin osoitin osoittamaan varattua muistilohkoa.



Osoittimien käyttöä

- int **p;
  int *q,r;
  int i;

  i= **p



q = &i; /* i:n osoite q:n arvoksi
i = *q+1; i = ++*q; /* i=6*/
i = *q++; /* ???? */
r = q; *r = 3; /* i=3 */

```
const int *p;
int const *p;
const int const *p;
```

```
char viesti [] = "On aika";   viesti: On aika\0
char *pv ="On aika";          pv: On aika\0
```

```
void *p;  i= *(int*) p;
```


Osoitin parametrina

- C:ssä vain arvoparametreja => funktio käyttää vain kopioita eikö voi mitenkään muuttaa saamiensa parametrien arvoja:

```
void swap(int x, int y) {
  int apu;
  apu=x;
  x=y;
  y= apu;
}
```

kopioita

x:n osoite, y:n osoite

```
void swap(int *x, int *y) {
    int apu;
    apu=*x;
    *x=*y;
    *y= apu;
}
```

Kutsu: swap (&x, &y);

x 3
y 4

```
double product (const double block, int size);
```

Varmistaa, ettei funktio muuta viiteparametrina saamaansa lohkoa

C: funktio-osoittimet

Java: "overriding"

- Esimerkki: funktio voi suorituksen aikana vaihtaa käyttämäänsä lajittelualgoritmia alkioiden lukumäärän perusteella

```
int (*fp) (void);
```

Osoitin int-tyypin palauttavaan funktiooon

```
int *fp()
```

Funktio, joka palauttaa int-tyyppisen osoittimen!

```
int fname(); /* kunhan funktio on oikean tyyppinen */
fp = fname;  /* fp() merkitsee nyt samaa kuin fname() */
```

- Funktion parametrina funktio

```
void qsort(*line[], int left, int right, int (*comp)(void *, void*));
```



```
void main (void) {
int choice;  double x, fx;
funcptr fp;
.....
funcprt function[7] = {NULL, sin, cos, tan, log , log_2, exp}; /*määriteltyjä funktioita*/
/*  funktiomenun tulostus: käyttäjää valitsee  haluamansa vaihtoehdon  */
 ....
scanf ("%i", &choice);
/* lisäksi tarkistetaan, että valinta on sallittu arvo */
...
if (choice ==0) break;
printf("Enter x: "); scanf/("%lg", &x);
fp = function[choice];
fx = fp(x);
printf("\n  (%g) = %g\n", x, fx);
}
}
```

```
typedef  double (*funcptr) (double );
```

function [0] [1] [2] [3] [4] [5] [6]
NULL sin cos tan log log_2 exp

Kurssikokeesta

- 20.10. klo 16-19
- Kokeessa saa olla mukana A4:n kokoinen muistilappu

Vähän kokeesta: mitä pitää osata

- Aiempien kurssin asioista perusohjelmointi, algoritmien kirjoittaminen, tietorakenteiden käyttö(taulukko, lista, pino) ja niiden tavanomaiset käsittelyrutiinit (lisääminen, poisto ja järjestäminen)
- C-kielen syntaksi ja semantiikka. Kirjastorutiinien käyttöä ei sinänsä edellytetä, mutta erimerkiksi merkkijonojen ja tiedostojen käsittelyyn käytettävät tavanomaiset funktiot on syytä hallita.
- Kielen rakenteista on hyvä hallita ainakin:
  - Funktioiden ja niiden parametrien käyttö
  - Taulukko
  - Tekstitiedosto
  - Linkitetty tietorakenne ja osoittimet
  - Komentoriviparametrit
  - Merkkijonot

Lisää kokeesta

- Näistä osattavista asioista muodostetaan sitten kokeessa erilaisia yhdistelmiä eri tehtävissä.
- Esimerkiksi toukokuun 2005 kokeessa oli
  - tehtävässä 1:
    - funktion käyttöä ja tietojen lukemista käyttäjältä (eli tiedostosta stdin),
  - tehtävässä 2:
    - funktio, osoittimia ja linkitetyn listan kopiointi ja järjestäminen
  - tehtävässä 3:
    - Tekstitiedosto, komentoriviparametri ja taulukko
- Koska koeaikaa on vain 2,5 tuntia, niin kolmeen tehtävään vastaaminen edellyttää jonkinlaista rutiinia C-ohjelmien kirjoittamisessa
  - Tätä rutiinia on saatu kurssin harjoitustehtäviä tehdessä!
  - Lunttilappu helpottaa asioiden muistamista!