

Kertausta

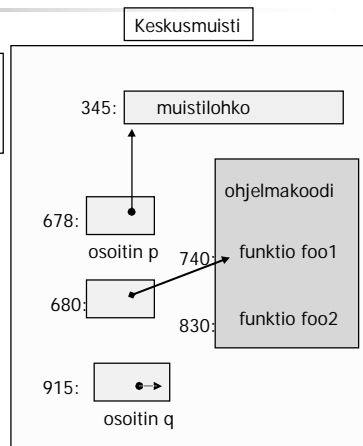
Luento 10
13.10.2006

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 muistilohkelle sekä  
asettavat osoittimen osoittamaan ko. muistilohkoa.
```

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

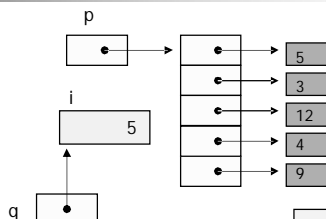


Osoittimien käyttöä

```

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

  i= **p
  
```



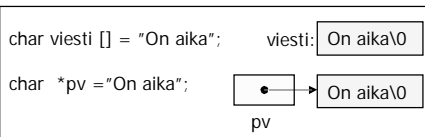
```

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;
  
```

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



C-ohjelmointi
Syksy 2006

3

Osoitin parametrina

x:n osoite, y:n osoite

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

```

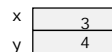
void swap(int x, int y) {
  int apu;
  apu=x;
  x=y;
  y= apu;
}
  
```



```

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

Kutsu: swap (&x, &y);



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

Varmistaa, ettei funktio muuta viiteparametrina saamaansa lohkoa

C-ohjelmointi
Syksy 2006

4

C: funktio-osoittimet

- Esimerkki: funktio voi suorituksen aikana vaihtaa käyttämänsä lajittelualgoritmia alkioden lukumäärän perusteella

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

```
int *fp()
```

Funktio, joka palauttaa int-tyyppisen osoittimen!

Osoitin int-tyypin palauttavaan funktioon

```
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: käyttäjä 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

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

Vähän kokeesta: mitä pitää osata

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



Lisää kokeesta

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