C-kielen tyyppijärjestelmä

Seminaarityö

Julius Meriläinen

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos
Ohjelmointikielten periaatteet 2004
Helsinki 21.4.2004

Contents

Chapter 1

Johdanto

1.0.1 Historia

Dennis Ritchie suunnitteli C-kielen Bell Laboratories:lla 1970-luvun alkupuolella. C-kielen esi-isiksi ja vaikuttajiksi voidaan lukea ALGOL, CPL ja BPCL ja B-kieli. C on yleiskäyttöinen ohjelmointikieli jota on perinteisesti käytetty systeemiohjelmointiin [HaSt02], [Keri88].

1.0.2 Standardit

Alkuperäisen C-kielen kuvaus on kirjassa *The C Programming Language, Brian W. Kernighan, Dennis M. Ritchie, 1978.*
Kirjallisuudessa usein käytetty lyhennys tälle kuvaukselle on "K&R" ja siitä käytetään nimitystä perinteinen C-kieli. Vuonna 1982 American National Standards Institute (ANSI) muodosti komitean X3J11, jonka tehtävänä oli luoda standardi C-kielelle ja sen kirjastorutiineille. Komitean tuotos hyväksyttiin vuonna 1989 nimellä *American National Standard X3.159-1989*, joka tunnetaan paremmin nimellä "ANSI C" [HaSt02].

Kansainvälisellä areenalla International Standards Institute (ISO) muodosti teknisen komitean JTC1/SC22/WG14 jonka tehtävänä oli muodostaa ANSI C standardista kansainvälinen standardi. Tämän standardin nimeksi tuli *ISO/IEC 9899:1990*, joka on olennaisilta osiltaan identtinen X3.159 standardin kanssa. X3.159 ja ISO 9899:1990 standardeja kutsutaan yhteisnimityksellä standardi C:ksi tai C89 [HaSt02], [Pla92].

Vuonna 1995 C89 standardiin tehtiin muutama muutos. Tätä tuotosta kutsutaan nimellä "C89 with Amendment 1" tai C95 [HaSt02].

Vuonna 1999 tuli C-kieleen uusi standardi joka laajensi aiempaa standardia, korvaten tämän. Tämän uuden standardin nimeksi tuli *ISO/IEC 9899:1999* tai C99. C99 tuo lisää monia uusia piirteitä C-kieleen ja sen

kirjastoihin [HaSt02].

Chapter 2

Tyypit

C-kieli tarjoaa suuren määrän sisäänrakennettuja tyyppejä, erikokoisia kokonaislukuja, liukulukuja, osoittimia, enumeraatioita, taulukoita, tietueita, unioneja ja funktioita [HaSt02]. Ohessa annamme lyhyen kuvauksen jokaisesta luetellusta tyypistä.

2.0.3 Kokonaislukutyypit

Kokonaisluvut jaetaan C-kielessä, etumerkillisiin-, etumerkittömiin-, boolean- ja merkkityyppeihin [HaSt02].

C-kielen etumerkillisiä kokonaislukutyyppejä ovat *short, int, long ja long long*. Etumerkilliset tyypit voidaan ilmaista joko etuliitteellä *signed* tai ilman. Standardi C määrittelee minimi koot kokonaislukutyypeille. Tyypin char täytyy olla vähintään 8 bittiä leveä, tyypin short 16 bittiä, tyypin long 32 bittiä ja tyypin long long 64 bittiä. Yleisesti käytössä oleva etumerkillisten kokonaislukujen esitystapa on kahdenkomplementti, jossa kokonaisluvun arvot ovat väliltä -2^{n-1} ... $2^{n-1}-1$, missä n on kokonaisluvun leveys.

```
/* Esimerkkejä; */
short   i, j;
int     a;
signed  b;  /* pelkkä signed on ekvivalentti
               tyypin signed int ja int kanssa */
```

Jokaiselle etumerkilliselle tyypille on vastaava etumerkitön tyyppi, jotka ovat yhtä leveitä, mutta käyttävät erilaista kokonaisluku koodausta. Etumerkitön tyyppi ilmaistaan lisäämällä etuliite *unsigned* vastaavan etumerkillisentyypin eteen. Etumerkittömän kokonaisluvun arvot ovat välillä 0 ... 2^n-1, missä n on kokonaisluvun leveys.

```
/* Esimerkkejä; */
```

```
unsigned short  i, j;
unsigned int    a;
unsigned        b;  /* pelkkä unsigned on ekvivalentti
                       tyypin unsigned int kanssa */
```

Merkkityyppi *char* on kokonaisluku tyyppi C-kielessä [HaSt02]. Merkkityyppejä on kolmea erilaista, etumerkillistä, etumerkitöntä ja tavallista (signed char, unsigned char, char). Jokainen näistä merkkityypeistä vie samanverran tilaa, mutta edustaa eri arvoja. Etumerkitöntä ja etumerkillistä esitystä käytetään kuten vastaavia muita kokonaislukutyyppejä. Tavallinen merkkityyppi (char) voi olla joko etumerkillinen tai etumerkitön riippuen toteutuksesta.

```
/* Esimerkkejä; */
unsigned char   i, j;
signed char     a;
char            b;
```

Boolean tyyppi *__Bool* on tullut C-kieleen C99 standardin mukana, se on etumerkitön kokonaislukutyyppi joka voi saada vain arvot 0 tai 1 [HaSt02].

2.0.4 Liukuluktyypit

C-kielen liukulukutyypit ovat *float, double, long double* sekä C99:n mukana tulevat kompleksiset liukulukutyypit *_Complex, _Imaginary*, jotka sisältävät etuliittenään joko float, double tai long double liiteen [HaSt02].

```
/* Esimerkkejä; */
double          i, j;
long double     a;
float _Complex  b;
```

2.0.5 Osoitintyypit

Jokaiselle tyypille T voidaan muodostaa osoitintyyppi [HaSt02]. Osoitintyypit ovat joko objekti- tai funktionosoittimia, riippuen onko T objekti vai funktio. Osoitintyypin arvo on objektin tai funktion osoite. Osoitintyyppiin sovellettavia operaatioita ovat mm. sijoittaminen, vähennys. Osoitintyypin koko riippuu toteutuksesta, dataan viittaavat osoittimet voivat olla lyhyempiä tai pidempiä kuin funktioihin viittaavat osoittimet. Usein osoitintyyppien koolla on yhteys kokonaislukujen kokokoon (tyyppi long voi olla yhtä leveä kuin osoitintyyppi), mutta näin ei välttämättä tarvitse olla.

```
Esimerkkejä osoitintyypeistä:
int *ip        /* ip on osoitin objektiin jonka tyyppi on int       */
long (*fp)();  /* fp on osoitin funktioon joka palauttaa arvon long */
```

2.0.6 Taulukkotyypit

Taulukot ovat sarja peräkkäisiä elementtejä tyyppiä T [HaSt02].

```
Esimerkki taulukkotyypistä:
int taulu[10];
```

C-kielessä osoitintyypeillä ja taulukoilla on yhteys. Kun taulukkotyyppi esiintyy lausekkeessa niin se konvertoidaan tyypistä "taulukko tyyppiä T" muotoon "osoitin taulukon tyyppiä T ensimmäiseen alkioon".

```
int a[10], *ip;
ip = a;  /* ip osoittaa taulukon a ensimmäiseen alkioon */
ip = &a[0]; /* yllä oleva on täsmälleen sama tämän kanssa */
```

Taulukon alkioiden indeksointi määritellään myös osoitin aritmetiikan avulla. Ilmaus a[i] (taulukon a, i:s alkio) on vastaava kuin ilmaus *((a) + (i)), missä a on konvertoitu muotoon &a[0]. Vastaavasti indeksointi a[i] voidaan myös ilmaista muodossa i[a].

Uutena piirteenä C-kieleen on C99:n mukana tullut muuttuvan kokoiset taulukot, jotka ovat taulukoita joiden koko saadaan selville vasta ajoaikana [HaSt02].

```
void fun(int i)
{
   int a[i];  /* i:n arvo saadaan vasta ajoaikana */
   ...
}
```

2.0.7 Enumeraatiotyypit

Enumeraatiotyyppi on joukko kokonaislukuarvoja, jotka tunnus "enumeraatiovakio" ilmaisee [HaSt02]. Enumeraatiovakioiden tyyppi on *int*, jonka koko on toteutuksesta riippuva.

```
enum tmp {eka, toka, kolmas};
/* nimi: arvo,  eka: 1, toka: 2, kolmas: 3 */
```

2.0.8 Tietuetyypit

Tietuetyypi on kokoelma nimettyjä komponentteja [HaSt02]. Tietueet eivät voi sisältää ilmentymiä itsestään, mutta osoitin ilmentymään itsestä on sallittu.

```
/* Määrittelemme tietueen jolla on kaksi kokonaisluku-
   komponenttia ja kaksi liukulukukomponenttia
   sekä osoitin ilmentymään itsestä */

struct koe {
  int    a, b;
  double c, d;
  struct koe *next;
};
```

C-kieli tarjoaa mahdollisuuden kokonaislukujen pakkaamisen pienempään tilaan kuin mitä ne normaalisti veisivät tilaa [HaSt02], tämä on mahdollista toteuttaa tietueiden avulla.

```
/* Määrittelemme tietueen jonka ensimmäinen alkio vie
   tilaa kolme bittiä ja jälkimmäinen viisi bittiä. */

struct koe2 {
  int a : 3;
  int b : 5;
};
```

2.0.9 Unionityypit

Jokaiselle komponentille unionityypissä varataan tilaa alkaen unionityypin alusta. Unionityyppi voi pitää sisällään vain yhden komponenttinsa arvon kerrallaan [HaSt02]. Unionityypin koko on sen suurimman komponentin vaatima tila sekä mahdollinen täyte ryhmittämisrajalle.

```
/* Määrittelemme unionin jolla on kolme komponenttia */
union un {
  long  a;
  char  b[2];
  int   c;
};
```

2.0.10 Funktiotyypit

Tyyppi "funktio paluttaa T:n", missä T on mikä tahansa tyyppi paitsi taulukko tai funktio, kutsutaan funktiotyypiksi [HaSt02]. C-kielessä funktio ei voi palauttaa taulukkotyyppiä eikä funktiota, mutta osoitin taulukkoon ja funktioon voidaan palauttaa.

```
int fun(int a)
```

```
{
  return a*a;
}
```

2.0.11 void tyyppi

Tyypillä *void* ei ole arvoa eikä siihen voida soveltaa mitään operaatioita [HaSt02]. Void tyyppiä käytetään lähinnä funktion paluu tyyppinä, merkitsemään että funktio ei palauta mitään arvoa. Tyyppimuunnoksella ilmaistaan void tyypillä, että emme ole kiinnostuneita paluuarvosta. Geneerisenä osoittimena on käytössä *void **. Funktion parametrina voi olla tyyppi void ilmaisemassa, että funktiolla ei ole argumentteja [Rat03].

```
Esimerkkejä:

(void)printf(‘‘Hello, World!\n’’);
/* emme välitä printf kirjastorutiinin paluuarvosta */

void fun(void);
/* funktio ei palauta mitään arvoa,
   eikä sillä ole yhtään argumenttia */
```

2.0.12 typedef tyypit

typedef määrittää tunnisteen joka nimeää typpejä, se ei luo uusia tyyppejä, ainoastaan synonyymeja muille tyypeille [KeRi88].

```
Esimerkkejä:

typedef int *IP;
typedef int A5[5];
typedef unsigned long size_t;

käyttö:
IP ip;  /* ip on osoitin kokonaislukuun */
A5 a5;  /* a5 on viiden elementin taulukko */
size_t b; /* b on unsigned long tyyppiä */
```

LÄHTEET:

[HaSt02] Samuel P. Harbison III, Guy L. Steele Jr.:
C A Reference Manual, Fifth Edition, Prentice Hall, 2002.

[KeRi88] Brian W. Kernighan, Dennis M. Ritchie.:
THE C PROGRAMMING LANGUAGE, Second Edition,
Prentice Hall, 1988.

[Pla92] P. J. Plauger.:
THE STARDARD C LIBRARY, Prentice Hall, 1992.

[Rat03] Rationale for International Standard– Programming Languages–
C, Revision 5.10, April-2003.