Sintaksa C jezika

Varijable i konstante

Tipovi podataka

- Tip podatka određuje skup vrijednosti koje varijabla može poprimiti i operacije koje se nad njom mogu izvoditi. C definira sljedeće osnovne tipove podataka:

Tip	Opis
`char`	znak / 8-bitni cijeli broj (`unsigned`)
`short int`	najmanje 16-bitni cijeli broj (`unsigned`)
`int`	najmanje 16, 32 ili više-bitni cijeli broj (`unsigned`)
`long int`	najmanje 32-bitni cijeli broj (`unsigned`)
`float`	realni broj jednostruke preciznosti (najčešće 32 bita)
`double`	realni broj dvostruke preciznosti (najčešće 64 bita)
`long double`	realni broj proširene preciznosti (najčešće 64 bita ili više)

Stvarne duljine pojedinih tipova podataka nisu standardizirane i ovise o procesoru za koji je pisan prevodioc.

Imena varijabli

- Program dohvaća vrijednost varijable preko njenog imena.
- Ime varijable mora ispunjavati sljedeće uvjete:
  1. Mora biti legalni identifikator, tj. niz ASCII znakova koji počinje slovom (najmanje 31 znak je značajan),
  2. Ne smije biti ključna riječ,
  3. Mora biti jedinstveno u području dosega varijable.
- Konvencija: Imena varijabli i funkcija započinju malim slovom. Za imena konstanti koriste se velika slova. Ako ime varijable sadrži više riječi, riječi su povezane i svaka riječ osim prve počinje velikim slovom. Znak "_" može se koristiti za odvajanje riječi, ali se po konvenciji koristi samo za razdvajanje riječi u imenu konstante.

- Ključne riječi (*keywords*):

`auto`	`double`	`int`	`struct`
`break`	`else`	`long`	`switch`
`case`	`enum`	`register`	`typedef`
`char`	`extern`	`return`	`union`
`const`	`float`	`short`	`unsigned`
`continue`	`for`	`signed`	`void`
`default`	`goto`	`sizeof`	`volatile`
`do`	`if`	`static`	`while`

- C-ima 32 ključne riječi, za razliku od jezika više razine koji ih imaju preko 100. U C-u se većina operacija rješava **funkcijama**.

- Primjeri imena varijabli:

```
godinaRodjenja
jedinstveniMaticniBroj
temperatura
i1
. . .
```

- Za imena pomoćnih varijabli (npr. indeksi petlji) uobičajeno se koriste slova `i`, `j`, `k`, `m`, `n`, po potrebi u kombinaciji s brojevima. Slovo `l` nije preporučljivo koristiti zasebno jer ga je lako zamijeniti s brojkom `1`.

- Primjeri deklaracije varijabli:

```
int i;
float temperaturaZraka;
float x, y=5.8;
char c = 'A';
```

- Napomene:
  - Jednom deklaracijom moguće je obuhvatiti više varijabli, čija imena treba odvojiti zarezom.
  - Kod deklaracije varijablu je moguće i inicijalizirati. Uz vrijednost moguće je definirati i tip podatka, koji ovisi o tipu varijable i opcionalnom sufiksu koji može biti f, F, l, L, u ili U.

Konstante

- Navodi se ključna riječ `const` prije tipa, npr:

```
const double PI = 3.14159265359;
```

- Cjelobrojne konstante mogu se osim u dekadskom, navoditi u oktalnom ili heksadecimalnom sustavu:

```
125 = \o175 = \x7D
```

- Znakovne konstante mogu se navoditi na sljedeće načine:

```
char c = 'A', d = 55, e = '\n';
```


Doseg varijable

- Doseg varijable (eng. *scope*) definiran je kao dio programa u kojem je varijablu moguće dohvatiti preko njenog imena.
- Mjesto deklaracije varijable određuje njezin doseg. Prema mjestu deklaracije varijable dijelimo na lokalne i globalne.
  - Lokalne varijable deklariraju se unutar bloka naredbi i imaju doseg od mjesta deklaracije do kraja bloka:

```
naredba {
        ...
        int i;
        ...
}
```

  - Globalne varijable deklariraju se izvan bloka, dakle i izvan funkcija.

Sintaksa C jezika

Operatori

- C podržava aritmetičke operatore za cijele i realne brojeve:

Operator	Upotreba	Opis
+	op1 + op2	zbraja op1 i op2
-	op1 – op2	oduzima op2 od op1
*	op1 * op2	množi op1 i op2
/	op1 / op2	dijeli op1 sa op2
%	op1 % op2	ostatak dijeljenja op1/op2

Napomene:

- Operator % definiran je samo za cijele brojeve.
- Operator – koristi se i kao unarni operator predznaka.

- Ako su operandi različitog tipa, tip rezultata određuje se primjenom sljedećih pravila:
  - Ako je jedan od operanada `long double`, i drugi se pretvara u `long double`,
  - inače, ako je jedan od operanada `double`, i drugi se pretvara u `double`,
  - inače, ako je jedan od operanada `float`, i drugi se pretvara u `float`,
  - inače, `char` i `short` pretvaraju se u `int`,
  - te ako je jedan od operanada `long`, i drugi se pretvara u `long`.

- Dva skraćena unarna operatora za uvećanje i smanjenje vrijednosti varijable za 1 su ++ i `--`:

Operator	Upotreba	Opis
`++`	`op++`	Uvećava `op` za 1; Vraća vrijednost `op` prije uvećavanja.
`++`	`++op`	Uvećava `op` za 1; Vraća vrijednost `op` poslije uvećavanja.
`--`	`op--`	Umanjuje `op` za 1; Vraća vrijednost `op` prije umanjivanja.
`--`	`--op`	Umanjuje `op` za 1; Vraća vrijednost `op` poslije umanjivanja.

Poredbeni operatori i logički uvjeti

- Poredbeni operatori uspoređuju dva operanda i vraćaju logičku vrijednost ovisno o rezultatu usporedbe:

Operator	Upotreba	Vraća `true` ako:
`>`	`op1 > op2`	`op1` je veći od `op2`
`>=`	`op1 >= op2`	`op1` je veći ili jednak `op2`
`<`	`op1 < op2`	`op1` je manji od `op2`
`<=`	`op1 <= op2`	`op1` je manji ili jednak `op2`
`==`	`op1 == op2`	`op1` i `op2` su jednaki
`!=`	`op1 != op2`	`op1` i `op2` su različiti

- Operandi `op1` i `op2` mogu biti bilo kojeg osnovnog tipa.

- Logički uvjeti koriste se za formiranje složenih logičkih izraza:

Operator	Upotreba	Vraća `true` ako:
`&&`	`op1 && op2`	`op1` i `op2` su `true`, `op2` se ispituje samo ako je `op1 true`
`||`	`op1 || op2`	`op1` ili `op2` su `true`, `op2` se ispituje samo ako je `op1 false`
`!`	`!op`	`op` je `false`

Operatori posmaka i logički operatori

- Operatori posmaka pomiču bitove prvog operanda desno ili lijevo (broj mjesta posmaka određuje drugi operand):

Operator	Upotreba	Opis
>>	`op1 >> op2`	pomiče bitove op1 u desno za op2 mjesta
<<	`op1 << op2`	pomiče bitove op1 u lijevo za op2 mjesta

- Logički operatori obavljaju logičke operacije nad pojedinim bitovima operanada:

Operator	Upotreba	Opis
&	`op1 & op2`	logička operacija I
|	`op1 | op2`	logička operacija ILI
^	`op1 ^ op2`	logička operacija EX ILI
~	`~op`	komplement

Operatori pridruživanja

- Osnovni operator pridruživanja je "=". Pridružuje varijabli s lijeve strane znaka jednakosti vrijednost varijable, konstante ili izraza s desne strane.
- C omogućuje korištenje skraćenih operatora koji kombiniraju aritmetičku, logičku ili operaciju posmaka i operaciju pridruživanja:

```
i = i + 5;
```

možemo skraćeno pisati kao:

```
i += 5;
```

Tako pisani izrazi su pregledniji, npr.:

```
yyval[yypv[p3+p4] + yypv[p1+p2]] += 2;
```

je preglednije od:

```
yyval[yypv[p3+p4] + yypv[p1+p2]] = yyval[yypv[p3+p4]
  + yypv[p1+p2]] + 2;
```


Operator	Upotreba	Ekvivalentan dulji zapis
+=	op1 += op2	op1 = op1 + op2
-=	op1 -= op2	op1 = op1 – op2
*=	op1 *= op2	op1 = op1 * op2
/=	op1 /= op2	op1 = op1 / op2
%=	op1 %= op2	op1 = op1 % op2
&=	op1 &= op2	op1 = op1 & op2
|=	op1 |= op2	op1 = op1 | op2
^=	op1 ^= op2	op1 = op1 ^ op2
>>=	op1 >>= op2	op1 = op1 >> op2
<<=	op1 <<= op2	op1 = op1 << op2

Ostali operatori

Operator	Upotreba	Opis
`?:`	`op1 ? op2 : op3`	ako je `op1` `true` vraća `op2`, inače vraća `op3`
`[]`		deklariranje i stvaranje polja, dohvat elemenata
`. i ->`		dohvat elemenata strukture ili unije
`(parametri)`	`(p1,p2,...,pn)`	argumenti u pozivu metode
`(tip)`	npr. `(int)i`	pretvara vrijednost u zadani tip
`*`		pokazivač
`&`		adresa varijable

Prioritet izvođenja operatora

postfix operatori	`() [] -> .`
unarni operatori	`! ~ ++ -- + - * & (tip)`
multiplikativni	`* / %`
aditivni	`+ -`
posmaci	`<< >>`
relacijski	`< > <= >=`
jednakost	`== !=`
logički I	`&`
logički EX ILI	`^`
logički ILI	`|`
logički uvjet I	`&&`
logički uvjet ILI	`||`
uvjetni	`?:`
pridruživanje	`= += -= *= /= %= &= ^= |= <<= >>=`

Sintaksa C jezika

Komentari

Komentari

- C podržava dvije vrste komentara:

```
/* tekst komentara
(može se protezati kroz više redaka) */
```

  - Prevodioc ignorira sve što se nalazi između `/*` i `*/`

```
// tekst
```

  - Prevodioc ignorira sve što se nalazi između `//` i kraja retka

Sintaksa C jezika

Pseudonaredbe

Pseudonaredbe

- Pseudonaredbe su naredbe prevodiocu (ne pojavljuju se u izvršnom kodu).
- Najvažnije pseudonaredbe su:

```
#include <datoteka.h> ili #include "datoteka.h"

#define KONSTANTA vrijednost
```

Sintaksa C jezika

Naredbe za kontrolu toka programa

Uvod

- Dok ne naiđe na neku od naredbi za kontrolu toka programa, procesor izvodi naredbe slijedno, onim redom kojim su napisane u programu.

- Naredbe za kontrolu toka koriste se za uvjetno izvođenje grupe naredbi, za višestruko izvođenje grupe naredbi zadani broj puta ili do ispunjenja nekog logičkog uvjeta, odnosno uvijek kada je potrebno promijeniti slijedni tok izvođenja naredbi.

- Naredbe za kontrolu toka programa dijelimo u tri kategorije:

Kategorija	Naredba
petlje	`while, do-while , for`
odluke	`if-else, switch-case`
grananja	`break, continue, return, goto`

Naredbe `while` i `do - while`

- `while` petlja koristi se za kontinuirano izvođenje bloka naredbi dok je zadani logički uvjet istinit.

```
while (izraz) {
      naredbe;
}
```

- Naredba `while` na početku izračunava izraz, koji mora vratiti logičku vrijednost. Ako je izraz istinit, izvodi se pripadni blok naredbi. Ispitivanje izraza i izvođenja bloka naredbi ponavlja se sve dok je izraz istinit.
- Ukoliko logički izraz na početku nije istinit, pripadni blok naredbi neće se izvršiti niti jednom.

- `do - while` petlja slična je `while` petlji, ali se u ovom slučaju logički izraz ispituje na kraju.

```
do{
     naredbe;
} while (izraz);
```

- Pripadni blok naredbi uvijek će se izvršiti barem jednom.

Naredba `for`

- Naredba `for` omogućuje jednostavno izvođenje iteracija uz promjenu vrijednosti indeksa u zadanom rasponu brojeva.

```
for (inicijalizacija;terminiranje;inkrement) {
      naredbe
}
```

- `Inicijalizacija` sadrži izraz za inicijalizaciju petlje, koji se izvodi jednom kod ulaska u petlju. `Terminiranje` je logički izraz koji određuje trenutak završetka petlje. Izračunava se na početku svake iteracije i ako nije istinit petlja završava. Inkrement je izraz koji se poziva na kraju svake iteracije. Svi navedeni izrazi mogu se izostaviti, tako da npr. naredba `for ( ; ; )` daje beskonačnu petlju.

- Primjer:

```
for (i = 0; i<100; i++) {
      naredbe;
}
```

- Gornja petlja izvršiti će se 100 puta, a cjelobrojna varijabla `i` poprimati će vrijednosti 0, 1, 2, ..., 98, 99.
- Izraz za terminiranje petlje uspoređuje vrijednost indeksa sa zadanom graničnom vrijednosti, a dio za inkrement uvećava vrijednost indeksa na kraju svake iteracije.

Naredba `if / else`

- Naredba if omogućuje programu selektivno izvođenje grupe naredbi.

```
if (izraz) {
      naredbe;
}
```

- Ako je izraz istinit blok naredbi će se izvršiti, a u suprotnom se izvođenje programa nastavlja iza bloka.
- Ponekad se zahtijeva grananje programa na način da se ukoliko je uvjet ispunjen izvede jedan blok naredbi, a ako nije drugi blok.

```
if (izraz) {
      naredbe1;
} else {
      naredbe2;
}
```

- Operator `?:` je reducirana verzija `if` naredbe:

```
uvjet ? operand1 : operand2
```

- Primjer: program treba ispitati da li je cijeli broj pozitivan ili negativan i ispisati odgovarajuću poruku.

```
if (i >= 0) {
    printf("Broj %d je pozitivan.\n", i);
} else {
    printf("Broj %d je negativan.\n", i);
}
```

Uz korištenje `?:` operatora:

```
printf("Broj %d je %s\n", i,
    (i>=0 ? "pozitivan." : "negativan."));
```


Naredba `switch`

- Naredba `switch` omogućuje uvjetno izvršavanje grupa naredbi ovisno o vrijednosti kontrolne cjelobrojne varijable. Sljedeći primjer razvrstava dane u tjednu u radne dane i dane vikenda:

```
switch (dan) {
    case 1:
    case 2:
    case 3:
    case 4:
    case 5:  printf("Radni dan\n"); break;
    case 6:
    case 7:  printf("Vikend\n"); break;
    default: printf("Greška\n"); break;
}
```

Naredbe grananja

- C podržava četiri naredbe grananja:
  - `break`
  - `continue`
  - `return`
  - `goto`
- Naredba `break` terminira `switch`, `for`, `while` i `do-while` naredbe, uz ograničenje da može terminirati samo unutarnju petlju.
- Naredba `continue` preskače trenutnu iteraciju petlje, od mjesta gdje se naredba nalazi do kraja petlje.

- Naredba `return` koristi se za povratak iz funkcije. Ako funkcija vraća rezultat, izraz koji određuje rezultat stavlja se uz `return` naredbu:

```
return izraz;
```

- Naredba `goto` omogućuje skok na bilo koju naredbu unutar funkcije. Korištenje se ne preporučuje jer narušava strukturiranu formu programa.