Даворка Р. Јандрлић

Горан М. Лазовић

Милош М. Вучић

Програмирање

Универзитет у Београду
Машински факултет
Катедра за математику
Београд, 2021

ПРОГРАМИРАЊЕ

др Даворка Р. Јандрлић др Горан М. Лазовић

Милош М. Вучић

Универзитет у Београду
Машински факултет
Катедра за математику
Београд, 2021. година

- др Даворка Р. Јандрлић, доцент
  Машински факултет у Београду
- др Горан М. Лазовић, ванредни професор
  Машински факултет у Београду
- Милош М. Вучић, сарадник у настави
  Машински факултет у Београду

Рецензенти:
др Горан Воротовић, ван. проф., Машински факултет, Београд
др Јелена Граовац, доцент, Математички факултет, Београд

Издавач:
Универзитет у Београду
Машински факултет
Ул. Краљице Марије бр. 16, Београд
тел.(011) 3370-760
факс (011) 3370-364
www.mas.bg.ac.rs

За издавача:
Декан, др Радивоје Митровић, ред.проф.

Уредник:
др Милан Лечић, ред.проф.
Председник комисије за издавачку делатност
Машинског факултета у Београду

Тираж: 1000 примерака

Штампање I издања одобрила:
Комисија за издавачку делатност
Машинског факултета у Београду
и
Декан Машинског факултета
Одлуком бр. 31/2021
Од 29.09.2021. године

Штампа: ”Planeta-print”, 11000 Београд
www.planeta-print.rs

Београд, 2021. године

ИСБН-978-86-6060-094-5

Предговор

Књига Програмирање је првенствено намењена студентима 1. године Машинског факултета у Београду и покрива комплетно градиво које студенти треба да савладају на предмету Програмирање.

Свако поглавље покрива теоријске основе једне области и укључује велики број решених примера, задатака за вежбу и питалица на основу којих студенти могу да тестирају своје знање и разумевање области након читања.

Посебну захвалност дугујемо рецензентима др Горану Воротовићу, ванредном професору Машинског факултета у Београду, и др Јелени Граовац, доценту Математичког факултета у Београду на детаљном читању рукописа и корисним саветима.

Аутори ће бити захвални свима који укажу на евентуалне грешке и допринесу побољшању уџбеника у наредном издању.

Садржај

2 Изрази

Свака обрада података неизоставно укључује неко израчунавање, само израчунавање се даље описује формулом коју зовемо израз. У C језику израз је моћно средство за решавање проблема. Полазећи од константи и променљивих (елементарни изрази), њиховим комбиновањем са операторима формирају се сложенији изрази, овај поступак се даље може понављати. Оператори су основни елементи за формирање израза и оно што C језик издваја од осталих програмских језика је богата колекција оператора. Оператори језика C ће се уводити постепено (због обима) кроз сва поглавља. Овде ће бити описани основни оператори, аритметички оператори, оператори доделе и оператори инкрементирања и декрементирања.

2.1 Аритметички оператори

Сем оператора % (остатак при целобројном дељењу) који захтева целобројне операнде, операнди осталих аритметичких оператора могу бити било ког нумеричког типа (целобројни или разломљени). Симболи аритметичких оператора су уобичајени (видети табелу 2.1).

Приметимо да се исти симболи $+$, $-$ користе за означавање

Табела 2.1: *Аритметички оператори*

унарни	$+, -$
бинарни	$+, -, *, /, \%$

различитих оператора. Ово не представља проблем јер се из контекста може закључити да ли је реч о бинарним $+$, $-$ (сабирање/одузимање) или унарним $+$, $-$ (задржавање/промена знака).

У случају бинарних оператора, тип вредности аритметичког израза је једнак типу операнада ако су они истог типа, па је вредност израза 3/4 једнака 0 (целобројно дељење), док је вредност израза 3.0/4. (разломљено дељење) једнака 0.75. Поставља се питање, која је вредност аритметичког израза када су операнди различитог типа? Приступ овој проблематици у C језику је следећи, врши се (имплицитна) конверзија ужег у шири тип података, па је вредност израза 3/4. након конверзије једнака вредности израза 3./4. што резултује разломљеном вредношћу 0.75. Вредност целобројног дељења није очигледна када је неки од операнада негативан, нпр. вредност целобројног израза -9/7 зависи од тога на коју страну се врши одсецање. Ако је одсецање ка $-\infty$ вредност је -2, док је у случају одсецања ка 0 односно ка $+\infty$ вредност једнака -1. *C99* стандард утврђује одсецање ка 0. На исти начин, вредност израза $-9\%7$ није очигледна, зависно од тога колико је -9/7, може бити +5 односно -2. Стандардом *C99* се утврђује да је знак резултата остатка целобројног дељења једнака знаку левог операнда, па је вредност израза $-9\%7$ једнака -2.

2.1.1 Приоритет оператора

Поставља се питање, како се израчунава вредност израза 2+3*4, да ли се прво врши сабирање, а затим множење или обратно?
Јадан начин за отклањање двосмислености овог типа је увођење заграда (прво се израчунава вредност у заградама). Други начин је дефинисање приоритета оператора. Генерално, у C језику су унарни оператори вишег приоритета од бинарних.
У случају бинарних аритметичких оператора, мултипликативни оператори $*$, $/$, $\%$ су вишег приоритета од адитивних оператора $+$, $-$.

2.1.2 Асоцијативност оператора

Дефинисање приоритета оператора није довољно, вредност израза у коме фигуришу оператори истог приоритета није очигледна. Нпр. која је вредност израза 2-3+4?
Вредност датог израза је 3, ако се прво врши одузимање а затим сабирање (2-3)+4, односно -5, ако се прво врши сабирање а затим одузимање 2-(3+4). Двосмисленост овог типа се отклања дефинисањем асоцијативности (груписање) оператора. Унарни аритметички оператори су десно асоцијативни (груписање сдесна), док су бинарни аритметички оператори лево асоцијативни (груписање слева), па је вредност израза 2-3+4 једнака (2-3)+4 односно 3.

2.2 Оператор доделе

Једном када је израз израчунат његова вредност се најчешће (није обавезно) смешта као нова вредност неке променљиве.

Нека је `var` променљива и `expr` израз, конструкција:

```
var=expr
```

је у C језику израз додељивања (у другим програмским језицима је то наредба додељивања). Оператор = је оператор (просте) доделе. Израчунавање израза додељивања се одвија на следећи начин: израчунава се вредност израза са десне стране, израчуната вредност се затим додељује променљивој са леве стране. Ако је неопходно извршиће се конверзија израчунате вредности у тип променљиве са леве стране. Коначно, вредност израза додељивања је вредност која је додељена променљивој са леве стране. Можемо закључити да је тип израза додељивања једнак типу променљиве са леве стране оператора доделе.

```
int i = 5; /* i dobija vrednost 5 */
int j = i; /* j dobija vrednost 5 */
int l = 6 * i - j; /* l dobija vrednost 25 */
...
int i;
float f;
i = 65.32; /* i dobija vrednost 65 */
f = 345; /* f je sada 345.0 */
...
```

Оператор = има нижи приоритет од аритметичких оператора и десно је асоцијативан, па се израз: `i=j=k=0` израчунава као следећи израз: `i=(j=(k=0))`. Треба опрезно користити овакве конструкције јер свако појединачно додељивање може укључити имплицитну конверзију оригиналне вредности која даље фигурише у наредним додељивањима:

```
int i;
float f;

i=f=1.1f; // f=-1.1, i=-1
...
f=i=1.1f; // i=-1, f=-1.0
...
```

2.2.1 Бочни ефекат

Није уобичајено да оператор мења своје операнде, нпр. у изразу `i+j` оператор + не мења своје операнде i и j. Оператор = мења вредност свог левог операнда. Овај ефекат зовемо бочни (споредни) ефекат израза додељивања. Очигледно лева страна израза додељивања не може бити било шта. То нпр. може бити променљива.

2.2.2 Л-вредност

Бочни ефекат израза додељивања је промена вредности левог операнда, па су следећи изрази додељивања неисправни:

```
12 = i
i+j = 5
-i = j
```

Оператор = захтева тзв. Л-вредност (енг. *L value*) за свој леви операнд. Дакле Л-вредност је објекат који може бити леви операнд израза додељивања. Очигледно мора имати придружену меморијску локацију и при томе може мењати своју вредност.

2.2.3 Здружени оператори доделе

Ажурирања вредности променљиве у којима се постојећа вредност користи да би се добила нова вредност су веома честа. C језик уводи здружене операторе доделе тако да се нпр. уместо `i=i/2` може писати `i/=2`. Нека је $\oplus$ оператор тада је $\oplus =$ здружени оператор додељивања и $var \oplus = expr$ је замена за $var = var \oplus expr$.

Здружени оператори доделе су $+=$, $-=$, $*=$, $/=$, $\%=$... Има их укупно 10 и имају исти приоритет и исту асоцијативност као и оператор просте доделе $=$.

2.2.4 Оператори инкрементирања и декрементирања

У случају да се постојећа вредност променљиве увећава за 1 (нпр. бројачка променљива), тада се у C језику уместо `i+=1` уводи оператор инкрементирања `++` чиме се добија још компактнији запис. Очигледно реч је о унарном оператору. Оператор инкрементирања има две верзије:

```
++i // prefiksni operator inkrementiranja
j++ // postfiksni operator inkrementiranja
```

Наизглед једноставни оператори могу бити прави изазов за коришћење. Поред тога што као и сви оператори доделе имају бочни ефекат, овај оператор се може користити у две верзије: префиксној и постфиксној верзији. У обе верзије се вредност променљиве увећава за 1, разликују се само у делу шта се проглашава за вредност самог израза. Вредност израза `++i` је ажурирана вредност променљиве i (прво се вредност од i увећава

за 1, а затим користи), док је вредност израза i++ постојећа вредност променљиве i (прво се вредност од i користи, а затим увећава за 1). Аналогно се уводи оператор декрементирања -- који ће вредност израза (променљиве) умањити за 1.
Постификсни оператори инкрементирања и декрементирања имају виши приоритет у односу на унарне аритметичке операторе и лево су асоцијативни, док префиксни оператори инкрементирања и декрементирања имају исти приоритет као унарни $+$ и $-$ и десно су асоцијативни.

2.2.5 Конверзија типа и *cast* оператор

Када у изразу фигуришу операнди различитог типа извршиће се одговарајућа конверзија. У случају оператора додељивања вредност израза са десне стране се конвертује у тип вредности променљиве са леве стране. У осталим случајевима врши се конверзија ужег у шири тип.
Ако је неки од операнада (онај ширег типа) разломљеног типа, правила конверзије су редом:

- Ако је један од операнада типа `long double` вредност другог операнда се конвертује у тип `long double`.
- Ако је један од операнада типа `double` вредност другог операнда се конвертује у тип `double`.
- Ако је један од операнада типа `float` вредност другог операнда се конвертује у тип `float`.

У случају да су операнди целобројног типа, прво се врши тзв. целобројна промоција: операнди ужег типа од `int` (`char`, `short`) се конвертују у тип `int` (односно `unsigned int` у зависности од

тога да ли се вредности типа који се конвертује могу сместити у тип `int` или не). Даље се, ако су и даље операнди различитог типа, конверзија врши у складу са следећом хијерархијом целобројних типова:

```
int -> unsigned int -> long -> unsigned long -> long long
    -> unsigned long long
```

Приметимо да имплицитна конверзија означеног целобројног типа у неозначени може бити извор неочекиваних резултата. Разлог за ово је формат репрезентације означених целих бројева. Бит највеће тежине се користи за репрезентацију знака, па негативним вредностима одговарају велике неозначене вредности као у следећем примеру:

```
int i=-1;
unsigned u=0;
i<u // poredjenje neoznacenih vrednosti
    (kompajler najcesce prijavljuje upozorenje)!
```

Вредност горњег израза поређења је 0 (нетачно)! Слично тип променљиве са леве стране израза додељивања нема утицаја на само израчунавање израза:

```
int i=10000;
long j=i*i;
```

У горњем примеру тип израза `i*i` је `int` без обзира што ће се та вредност касније конвертовати у тип променљиве j. Са друге стране, могуће је да се у типу `int` не може репрезентовати вредност `i*i`, па ће доћи до одсецања. C језик допушта експлицитну

конверзију cast оператором:

```
(tip)izraz
```

па се претходни проблем може отклонити на следећи начин:

```
int i=1000;
long j=(long)i*i; // tip izraza (long)i*i je long
```

Како је реч о унарним операторима, вишег су приоритета од бинарних. У следећем примеру се илуструју ефекти унарних аритметичких оператора + и −. Унарни оператор задржавања знака + не мења вредност свог операнда, али му мења тип ако је операнд ужег целобројног типа од типа `int`.

```
#include <stdio.h>

int main(void) {
    short sh = -1;
    printf("short: %u bajta\n", sizeof(sh));
    printf("celobrojna promocija %2d na %u
        bajta\n", +sh, sizeof(+sh));
    printf("celobrojna promocija %2d na %u
        bajta\n", -sh, sizeof(-sh));
    return 0;
}
```

2.2.6 Оператор `sizeof`

Стандард C језика не утврђује величину меморијског простора за репрезентацију вредности појединих типова података.

Оператор `sizeof` враћа ту вредност.

```
sizeof(tip)
```

Као и сви унарни оператори, `sizeof` оператор је вишег приоритета од бинарних. Аргумент `sizeof` оператора може бити израз и тада се могу изоставити заграде, али тада треба водити рачуна о приоритету оператора. Нпр. израз `sizeof i+j` је једнак `(sizeof i)+j`.

2.2.7 Табела приоритета

Табела 2.2 приказује приоритет до сада наведених оператора.

Табела 2.2: *Приоритет и асоцијативност уведених оператора*

Приоритет	Назив и симбол	Асоцијативност
1	постфиксни $++$ $--$	лева
2	префиксни $++$ $--$ унарни $+$ $-$ (tip) $sizeof$	десна
3	мултипликативни: $*$ / $\%$	лева
4	адитивни: $+$ $-$	лева
5	додељивања: $=$ $+=$ $-=$ $*=$ $/=$ $\%=$	десна

2.2.8 Редослед израчунавања подизраза

Како стандард језика не покрива све детаље, остављен је простор за различите имплементације у којима треба бити обазрив. Такав је случај са редоследом израчунавања подизраза. Поставља се питање, приликом израчунавања вредности израза

`(a+b)*(c-d)`, да ли се прво израчунава вредност левог подизраза `(a+b)` или десног подизраза `(c-d)`?
Све док подизрази немају бочних ефеката, редослед израчунавања није битан. Ако то није случај као у следећем примеру:

```
a = 5;
c = (b = a + 2) - (a = 1);
```

није очигледно шта ће бити вредности променљивих b и c. Уколико се прво израчунава леви подизраз `(b = a + 2)`, b добија вредност 7, а c добија вредност 6, у супротном ако се прво израчунава десни подизраз `(a = 1)`, b добија вредност 3, а c добија вредност 2. Неки компајлери могу да препознају овакве ситуације и да пријаве упозорење. Да би се овакви проблеми избегли треба избегавати коришћење оператора доделе у подизразима, увек је боље направити низ одвојених додела, па би једна верзија изгледала:

```
a = 5;
b = a + 2;
a = 1;
c = b - a;
```

2.3 Наредба израза

Сваки израз у C језику који се завршава знаком ; постаје наредба израза.

```
izraz;
```

Наредба израза има смисла ако има бочних ефеката:

```
i++;
```

У противном ако нема бочних ефеката може се изоставити јер нема смисла:

```
i+1;
```

2.4 Питалице

1. Шта ће бити исписано након извршавања следеће секвенце програма:

```
int i = 3.5 + 4.5;
printf("%.1d\n", i);
```

А. Грешка компајлера Б. 7 В. 8 Г. 8.0

2. Шта ће бити исписано након извршавања следеће секвенце програма:

```
float c = 3.5 + 4.5;
printf("%d", (int)f);
```

А. 8.000000 Б. 7 В. 8 Г. 8.0

3. Шта ће бити исписано након извршавања следеће секвенце програма:

```
int i = -5;
int k = i%-4;
printf("%d\n", k);
```

А. Грешка компајлера Б. -1 В. 1 Г. 0

4. Шта ће бити исписано након извршавања следеће секвенце програма:

```
int i=-1;
+i;
printf("i=%d +i=%d", i, +i);
```

А. Грешка компајлера Б. `i=-1 +i=-1` В. `i=0 +i=1` Г. `i=1 +=1`

5. Шта ће бити исписано након извршавања следеће секвенце програма:

```
int a=1,b=2;
a=a++;
b=++b;
printf("%d %d", a,b);
```

А. 1 3 Б. 1 2 В. 2 2 Г. 2 3

6. Који од наведених оператора је `cast` оператор?

А. (tip) Б. cast() В. // Г. Ништа од наведеног

2.5 Задаци

Задатак 2.1 *Написати програм који сабира два цела броја задата са стандардног улаза.*

```
#include <stdio.h>

int main()
{
    int a, b, c;
    printf("Unesi prvi broj : ");
    scanf("%d", &a);
    printf("Unesi drugi broj : ");
    scanf("%d", &b);
    c = a + b;
    printf("%d + %d = %d\n", a, b, c);

   return 0;
}
```

Задатак 2.2 *Написати програм којим се размењују вредности два цела броја унета са стандардног улаза.*

```
#include <stdio.h>

int main()
{
    int a, b;
    int pom;
    printf("Unesi prvi broj : ");
    scanf("%d", &a);
```

```
      stara (neuvecana) vrednost */
   y = b++;

   printf("Posle : x = ++a; \na = %d\nx = %d\n", a, x);
   printf("Posle : y = b++; \nb = %d\ny = %d\n", b, y);

   return 0;
}
```

2.5.1 Задаци за вежбу

Задатак 2.6 *Написати програм који израчунава обим и површину правоугаоника, ако је познато да су странице правоугаоника паралелне координатним осама и са улаза се задају темена правоугаоника (доње лево теме и горње десно теме) својим координатама. Резултат приказати на две децимале.*

Задатак 2.7 *Написати програм који омогућава унос двоцифреног целог броја нпр. ab и једноцифреног c (где су a и b цифре првог броја), а затим формира троцифрене бројеве abc, acb и cab и штампа их на пољу ширине 5 десно поравнате један испод другог.*

Задатак 2.8 *Написати програм који омогућава унос два рационална броја (уносе се у форми: бројилац/именилац нпр. 3/7), сабира унесене бројеве и резултат приказује у виду разломка (бројилац/именилац).*

Задатак 2.9 *Написати програм који омогућава унос датума у форми $dd.mm.gggg$, затим унети датум штампа у форми $mm/dd/gggg$.*