Програмирање
Материјал - 1. недеља

Београд, 2016.

Садржај

1

Управљање током програма

Иако C има много оператора насупрот томе има ограничен број команди. До сада су анализиране само две команде: `return` и команде за формирање израза. Преостале команде спадају у једну од три категорије, у зависности од тога како утичу на редослед извршавања програма:

- **Наредбе избора.** Наредбе `if` и `switch` омогућавају програму да изабере један од путева извршавања од неколико могућих.
- **Итерационе наредбе.** Наредбе `while`, `do` и `for` подржавају итерације (петље-извршавање у циклусима одређене групе наредби).
- **Наредбе скока.** Наредбе `break`, `continue`, `goto` омогућавају безусловно скакање на неко друго место у програму. У ову категорију спада и `return` наредба.

Поред набројаних група наредби преостале су само здружене наредбе, које групишу неколико наредби у једну као и тзв. *null* наредба која нема никакав ефекат извршавања. Пре увођења наредби избора, неопходно је објаснити логичке изразе који се формирају у оквиру наредби избора где се тестира њихова вредност. Логички изрази се формирају од релационих оператора ($<, <=, >, >=$), оператора једнакости ($==, !=$) и логичких оператора ($\&\&, ||, !$).

1.1 Логички изрази

Неке наредбе у C–у укључујући `if` наредбу тестирају услов у оквиру наредбе и проверавају да је вредност израза `true` или `false`. Пример поређења је `i < j`, треба напоменути да у другим програмским језицима резултат оваквог поређења може бити типа "boolean" или "logical" док је у C-у оваквом изразу придружена целобројна вредност 0(false) или 1(true).

1.2 Релациони оператори

Релациони оператори су исти као у математици $<, <=, >, >=$ само што је резултат њиховог извршавања 1 или 0. Нпр. израз `10 < 11` има вредност 1. Релациони оператори могу да се примењују над целобројним и реалним операндима. Такође је дозвољено да операнди буду различитог типа: `1 < 2.5` има вредност 1, док `4 > 5.6` има вредност 0. Израз `i < j < k` је дозвољен у C-у и еквивалента је изразу `(i < j)< k` јер су релациони оператори лево асоцијативни. Дакле у овом примеру се не проверава да ли се вредност променљиве i налази између вредности променљивих j и k.

1.3 Оператори једнакости

Оператори једнакости $(==, !=)$ су као и релациони оператори лево асоцијативни и резултат примене ових оператора даје вредности 0 или 1. Оператори једнакости имају нижи приоритет у односу на релационе операторе:

```
i < j == j < k
```

Овај израз је еквивалентан изразу:

```
(i < j) == (j < k)
```

1.4 Логички оператори

Сложенији логички изрази могу да се направе укључивањем и логичких оператора: and (&&), or (||) и not(!) . Прва два оператора су бинарна док је оператор негације унарни. Примена логичких оператора такође резултује вредностима 0 или 1. Операнди најчешће узимају вредности 0 или 1, али ово није неопходно напротив свака вредност различита од 0 се третира као тачно, док се нула третира као нетачно. Логички оператори се понашају на следећи начин:

- `!izraz` има вредност 1 ако *izraz* има вредност 0.
- `izraz1 && izraz2` има вредност 1 ако оба операнда-израза имају вредност 1.
- `izraz1 || izraz2` има вредност 1 ако било који од операнда-израза има вредност 1.

У свим другим случајевима је вредност наведених израза 0. Јако битна особина оператора коњукције и дисјункције је да они иводе лењо-израчунавање, у смислу да прво израчунавају вредности левог операнда а затим десног. Уколико се коначна вредност израза може закључити на овнову вредности левог операнда десни операнд-израз се не израчунава. Нпр.: `(i != 0)&& (j / i > 0)`. Ако је у овом изразу вредност променљиве i различита од 0, десна страна израза мора да се израчуна како би се утврдила коначна вредност израза. Уколико је вредност променљиве i једнака 0, израз `(j / i > 0)` се не извршава чиме се избегава дељење са 0. Треба бити јако пажљив са бочним ефектима у логичким изразима! Захваљујући лењом извршавању логичких оператора коњукције и дисјункције некада се бочни ефекти неће догодити. Нпр. у следећем изразу: `i > 0 && ++j > 0` Увећање променљиве j је бочни ефекат извршавања другог израза (операнда у коњукцији). Међутим ако је резултат извршавања првог израза `i > 0` false, израз `++j > 0` се не извршава а тиме се и не инкрементира променљива j. Проблем се може решити заменом места првог и другог израза `++j > 0 && i > 0` или још боље увећањем вредности променљивој j ван израза. Оператор негације има исти приоритет као унарни оператори + и -. Приоритет осталих логичких оператора је нижи од релационих и оператора једнакости. Нпр. `i < j && k == m` је исто што и `(i < j)&& (k == m)`. Оператор негације је десно асоцијативан док су остали логички оператори лево асоцијативни.

1.4.1 Наредба - `if`

Наредба - if омогућава управљане током програма (избор једног од више могућих путева), тестирањем израза у оквиру if наредбе. Најједноставнији облик ове наредбе је `if (izraz)naredba;`. Заграде око услова су обавезне оне су део if наредбе а не део израза. Када се if наредба извршава, израчунава се вредност израза у заградама: уколико је вредност израза различита од нуле извршава се *naredba* Напомена: не мешати операторе = и == они имају различито значење! Најчешће се ова наредба тестира да ли се променљива налази између неких вредности (у интервалу). `if (0 <= i && i < n)` За тестирање супротног услова користимо оператор дисјункције `if (i < 0 || i >= n)` У случају да је услов испуњен може постојати потреба да се изврши више од једне наредбе (здружене наредбе), у том случају је неопходно користити заграде: {*bloknaredbi*}. Груписањен наредби унутар заграда компајлеру се наглашава да се наредбе извршавају као једна.

1.4.2 Клаузула - `else`

```
if (izraz) naredba else naredba
```

Не постоји ограничење за израз који може да се појави у оквиру if наредбе, уствари могуће је и угњеждавати if наредбе. (Пример макс 3 броја). Угњеждавање је могуће до произвољне дубине. Равнање одговарајуће `if` и `else` гране омогућава лакше читање и разумевање програма. Стављање заграда чак и када нису потребне омогућава лакше разумевање кода и смањује могућност прављења грешке.

1.4.3 Каскадна `if` наредба

Често ћемо бити у прилици да тестирамо више услова и да се тестирање заврши чим је један од услова испуњен. За то је каскадна if наредба најбоље решење. (пример проверити да ли је вредност променљиве једнака, већа или мања нули)

1.4.4 Услововни изрази

`if` наредба омогућава програму да изврши једну од две акције у зависности од вредности израза који се проверава. C такође подржава оператор који омогућава изразу да изврши једну од две акције у зависности од вредности израза. Овај оператор се зове условни оператор и састоји се од симбола : и ? који се користе заједно у изразу: `izraz1 ? izraz2 : izraz3` Изрази 1, 2, 3 могу бити било ког типа. Условни оператор захтева три операнда па се на њега реферише и као на тернарни оператор.

1.4.5 `Boolean` тип података

Према стандарду C89 не постоји одговарајући логички тип података. Овај недостатак се једноставно превазилази увођењем целобројне променљиве којој се додељују вредности 1 или 0 и третирају као тачно односно нетачно. Такође је могуће увести макрое TRUE и FALSE. У новом стандардну C99 је уведен тип података `_Bool` (`<stdbool.h>`).

1.4.6 Наредба - `switch`

У случајевима када је потребно поредити вредност израза са више могућих вредности алтернатива `if` наредби је `switch` наредба.

```
switch (ocena) {
case 4: printf("Vrlo dobar");
      break;
case 3: printf("Dobar");
      break;
case 5: printf("Odlican")
      break;
case 1: printf("nedovoljan");
      break;
default: printf("nije korektna ocena");
      break;
}
```

Када се ова наредба изврши вредност променљиве оцена се пореди са вредностима 4,3,5,1. Ако се поклапа са 4 порука *Врло добар* се исписује и контрола тока се пребацује на прву наредбу после `switch` наредбе. Ако се вредност не поклопи ни са једном од наведених вредности `default` наредба се извршава и исписује се порука није коректна оцена. Ова наредба се једноставније чита од каскадне `if` наредбе, и бржа је што је посебно значајно када је више `case` израза. У оквиру `switch` наредбе треба да стоји целобројна вредност, карактери се третирају као целобројне вредности, а наредба није подржана за реалне вредности и стрингове. Свака ставка `case` почиње баш том кључном речју иза које стоји константан израз и не сме садржати променљиву (осим ако не представља макро) нити позиве функцијама. После ове наредбе може да се нађе произвољан број израза, уколико их је више није неопходно оградити их заградама. Дуплиране case наредбе нису дозвољене, редослед навођења у принципу није битан па тако ни `default` наредба не мора да буде на последњем месту. Само једна константа може да прати једну `case` наредбу, ипак неколико `case` израза може да претходи једној групи наредби. Нажалост не постоји начин да се напишу `case` наредбе које специфицирају интервал вредности. `default` наредба није неопходна.

1.4.7 Улога `break` наредбе

Улога `break` наредбе је да прекине извршавање `switch` блока и да програм крене са извршавањем прве наредбе након `switch`. Када се евалуира израз и крене са извршавањем одговарајућег case блока уколико `break` не постоји све наредбе се извршавају редом до краја `switch`. У следећем примеру ако је вредност израза *ocena* 3, биће одштампано *DobarOdlicannedovoljan* (извршавају се наредбе редом док се не наиђе на `break`).

```
switch (ocena) {
case 4: printf("Vrlo dobar");
      break;
case 3: printf("Dobar");

case 5: printf("Odlican");

case 1: printf("nedovoljan");
      break;
default: printf("nije korektna ocena");
      break;
}
```

Постоје ситуације када је згодно изоставити `break` наредбу:

```
int pozitivnih = 0; // promenljiva cuva broj pozitivnih ocena
    int ukupno = 0;   // racunaju se sve ocene
    switch (ocena) {
          case 5: case 4: case 3: case 2:
               pozitivnih++;
               // nema break - nastavlja se izvrsavanje (i uvecava broj
                   ukupnih)

               case 1:
               ukupno++;
}
printf("Prolaznost na ispitu je:%5.2f\n", 1.0*pozitivnih/ukupno*100);
```

1.5 Задаци са вежби

Задатак 1.1 *Написати програм који проналази максимум два цела броја унета са улаза.*

```c
#include <stdio.h>

int main() {
    int a, b;
    int max; // Vrednost maksimuma
    printf("Unesi prvi broj: ");
    scanf("%d",&a);

    printf("Unesi drugi broj: ");
    scanf("%d",&b);

    // Pretpostavimo da je prvi broj veci.
    max = a;

    // Ukoliko to nije slucaj, vrednost se azurira.
    if(max < b) {
      max = b;
    }

    printf("Maksimum je: %d\n", max);

    system("pause");
    return 0;
}
```

Задатак 1.2 *Написати програм који за унето* `x` *израчунава* `y` *по формули:*

$$y = \begin{cases} \frac{-x^3}{3} & za\ x > 0 \\ \frac{x+1}{2} & inače \end{cases}$$

```c
#include <stdio.h>
#include <math.h>
```

```
int main() {
    float x;
    float y;

    printf("x = ?\n");
    scanf("%f", &x);

    if(x > 0) {
         y = - pow(x, 3) / 3.0;
    }
    else {
         y = (x + 1) / 2;
    }

    printf("y = %f\n", y);

    system("pause");
    return 0;
}
```

Задатак 1.3 *Написати програм који проверава да ли се од задатих одсечака* a, b *и* c *може конструисати троугао. Ако може израчунати површину троугла, ако не, исписати одговарајућу поруку.*

```
#include <stdio.h>
#include <math.h>

int main() {
    double a, b, c;

    printf("Unesite stranice trougla a = ?, b = ?, c = ?\n");
    scanf("%lf%lf%lf", &a, &b, &c);

    if(a + b > c && a + c > b && b + c > a) {
         double s = (a + b + c) / 2;
         double P = sqrt(s * (s - a) * (s - b) * (s - c));

         printf("Povrsina trougla je: %.2g\n", P);
```

```
    }
    else {
        printf("Od unetih stranica ne moze se formirati trougao!\n");
    }

    system("pause");
    return 0;
}
```

Задатак 1.4 *Написати програм који проналази реална решења квадратне једначине задате у облику*

$$ax^2 + bx + c = 0$$

.

```
#include <stdio.h>
#include <math.h>

int main() {
    float a, b, c;
    float D; // diskriminanta

    printf("Unesite koeficijente kvadratne jednacine \
           a = ?, b = ?, c = ?\n");
    scanf("%f%f%f", &a, &b, &c);

    if(a == 0.0) {
        printf("Greska a = 0, jednacina nije kvadratna!\n");

        system("pause");
        return 1;
    }

    // Izracunava se diskriminanta
    D = b * b - 4 * a * c;
    if(D == 0.0) {
        printf("Jednacina ima jedinstveno resenje \
               x1 = x2 = %.2f\n", - b / (2 * a));
    }
```

```
    else if(D > 0.0) {
         printf("Jednacina ima dva realna resenja:\n");
         printf("x1 = %.2f\n", (-b - sqrt(D)) / (2 * a));
         printf("x2 = %.2f\n", (-b + sqrt(D)) / (2 * a));
    }
    else {
         printf("Jednacina nema realnih resenja.\n");
    }

    system("pause");
    return 0;
}
```

Задатак 1.5 *Написати програм који за задати редни број дана у недељи исписује његово име. У случају некоректног уноса исписати одговарајућу поруку.*

```
#include <stdio.h>

int main() {
    // Redni broj dana u nedelji.
    int dan;

    printf("Redni broj dana u nedelji:\n");
    scanf("%d", &dan);

    switch(dan) {
        case 1:
             printf("Ponedeljak\n");
             break;
        case 2:
             printf("Utorak\n");
             break;
        case 3:
             printf("Sreda\n");
             break;
        case 4:
             printf("Cetvrtak\n");
             break;
```

```
        case 5:
            printf("Petak\n");
            break;
        case 6:
            printf("Subota\n");
            break;
        case 7:
            printf("Nedelja\n");
            break;
        default:
            printf("Nekorektan unos, redni broj dana treba da \
                   je u intervalu [1, 7]\n");
    }

    system("pause");
    return 0;
}
```

Задатак 1.6 *Написати програм који симулира рад упрошћеног калкулатора. Са улаза се уноси цео број затим једна од аритметичких операција +, -, *, /, %, и на крају цео број, а затим се на стандарни излаз исписује вредност тог израза.*

```
#include <stdio.h>

int main() {
    int a, b; // operandi
    char op;  // operacija

    printf("Unesti izraz u obliku operand operacija operand:\n");
    scanf("%d%c%d", &a, &op, &b);

    switch(op) {
        case '+':
            printf("%d %c %d = %d\n", a, op, b, a + b);
            break;
        case '-':
            printf("%d %c %d = %d\n", a, op, b, a - b);
```

```
            break;
        case '*':
            printf("%d %c %d = %d\n", a, op, b, a * b);
            break;
        case '/':
            // Obratiti paznju na deljenje!
            printf("%d %c %d = %f\n", a, op, b, a / (b + 0.0));
            break;
        case '%':
            printf("%d %c %d = %d\n", a, op, b, a % b);
            break;
        default:
          // Obratiti paznju na ispis \ i %
          printf("Nekorektan unos, operacija mora biti jedna \
                 od: +, -, *, \\, %%\n");
    }

    system("pause");
    return 0;
}
```

1.5.1 Самосталан рад

Задатак 1.7 *Napisati program koji izračunava maksimum i minimum tri cela broja sa ulaza.*

Задатак 1.8 *Napisati program koji rešava linearnu jednačinu*

$$ax + b = 0$$

.

Задатак 1.9 *Ispitati da li se prave*

$$y = k_1 x + n_1$$

i

$$y = k_2 x + n_2$$

seku i ako se seku naći koordinate presečne tačke.

Задатак 1.10 *Napisati program koji za zadati redni broj meseca ispisuje broj dana u tom mesecu. Za mesec februar pretpostaviti da ima 28 dana.*