برنامه نویسی مقدماتی

مدرس : هادی حبیبی
ایمیل : h.habibi@irib.ir

تارنما:rlecture.ir

https://piazza.com/uast.ac.ir/fall2013/ce1/resources

جلسه پنجم

فهرست

- ساختارهای کنترل تکرار
  - تکرار کنترل شونده با شمارنده
  - تکرار کنترل شونده با نگهبان
  - تکرار کنترل شونده تو در تو
- عملگرهای انتساب
- عملگرهای افزایشی و کاهشی

پیاده سازی الگوریتم - مثال اول

- برای آشنایی با نحوه تبدیل یک الگوریتم به یک برنامه C در اینجا چند نمونه مساله برای محاسبه میانگین، را بیان می کنیم.
- مساله زیر را در نظر بگیرید :
  - کلاسی شامل ۱۰ دانشجو در یک کوئیز شرکت می کنند. نمرات این دانشجویان در اختیار شما قرار دارد. این نمرات اعداد صحیح بین ۰ تا ۱۰۰ می باشند. میانگین نمرات کلاس را پیدا کنید.
- الگوریتم حل این مساله اعداد نمرات را از ورودی دریافت می کند، میانگین آنها را محاسبه می کند و در نهایت نتیجه را در خروجی چاپ می کند.

پیاده سازی الگوریتم - مثال اول ...

- در اینجا از حلقه تکرار کنترل شونده با شمارنده استفاده می کنیم تا نمرات را از ورودی دریافت کنیم.
- در این تکنیک از یک متغیر که به آن شمارنده (counter) می گویند برای کنترل دفعات اجرای مجموعه ای از دستورات استفاده می شود.
- در این بخش شبه کد و برنامه C را نشان می دهیم.
- تکرار کنترل شونده با شمارنده، تکرار قطعی یا معلوم نامیده می شود چون در آن تعداد تکرار قبل از اینکه حلقه تکرار شروع شود مشخص می باشد.

پیاده سازی الگوریتم - مثال اول ...

```
1   Set total to zero
2   Set grade counter to one
3
4   While grade counter is less than or equal to ten
5       Input the next grade
6       Add the grade into the total
7       Add one to the grade counter
8
9   Set the class average to the total divided by ten
10  Print the class average
```

Fig. 3.5 | Pseudocode algorithm that uses counter-controlled repetition to solve the class average problem.

پیاده سازی الگوریتم - مثال اول ...

- ۱ مجموع نمرات را برابر صفر قرار بده.
- ۲ شمارنده تعداد نمرات را برابر ۱ قرار بده.
- ۳
- ۴ تا زمانی که شمارنده تعداد نمرات کوچکتر مساوی ۱۰ می باشد.
  - ۵ نمره بعدی را از ورودی بخوان.
  - ۶ این نمره را به مجموع نمرات اضافه کن.
  - ۷ یک واحد به مقدار شمارنده تعداد نمرات اضافه کن
- ۸
- ۹ مقدار میانگین کلاس را برابر تقسیم مجموع نمرات بر ۱۰ قرار بده.
- ۱۰ مقدار میانگین نمرات کلاس را چاپ کن.

پیاده سازی الگوریتم - مثال اول ...

```
1  /* Fig. 3.6: fig03_06.c
2     Class average program with counter-controlled repetition */
3  #include <stdio.h>
4
5  /* function main begins program execution */
6  int main( void )
7  {
8     int counter; /* number of grade to be entered next */
9     int grade; /* grade value */
10    int total; /* sum of grades input by user */
11    int average; /* average of grades */
12
13    /* initialization phase */
14    total = 0; /* initialize total */
15    counter = 1; /* initialize loop counter */
16
17    /* processing phase */
18    while ( counter <= 10 ) { /* loop 10 times */
19       printf( "Enter grade: " ); /* prompt for input */
20       scanf( "%d", &grade ); /* read grade from user */
21       total = total + grade; /* add grade to total */
22       counter = counter + 1; /* increment counter */
23    } /* end while */
```

Fig. 3.6 | C program and sample execution for the class average problem with counter-controlled repetition. (Part 1 of 2.)

پیاده سازی الگوریتم - مثال اول ...

```
24
25      /* termination phase */
26      average = total / 10; /* integer division */
27
28      printf( "Class average is %d\n", average ); /* display result */
29      return 0; /* indicate program ended successfully */
30   } /* end function main */
```

```
Enter grade: 98
Enter grade: 76
Enter grade: 71
Enter grade: 87
Enter grade: 83
Enter grade: 90
Enter grade: 57
Enter grade: 79
Enter grade: 82
Enter grade: 94
Class average is 81
```

Fig. 3.6 | C program and sample execution for the class average problem with counter-controlled repetition. (Part 2 of 2.)

پیاده سازی الگوریتم - مثال اول ...

- میانگین محاسبه شده در این مثال عدد ۸۱ می باشد.
- در واقع مجموع اعداد وارد شده برابر ۸۱۷ می باشد، که وقتی بر ۱۰ تقسیم می شود مقدار آن ۸۱.۷ می شود، از آنجاییکه میانگین عددی صحیح است بنابراین عدد محاسبه شده ۸۱ خواهد بود و بخش اعشاری حذف می شود.
- در بخش بعد می بینیم چگونه از اعداد اعشار استفاده کنیم.
- نکته » حتما متغیرهای مجموع و شمارنده را مقدار دهی اولیه کنید وگرنه نتایج محاسبه اشتباه خواهد شد.

پیاده سازی الگوریتم - مثال دوم ...

- بیایید مساله محاسبه میانگین نمرات را بدین شکل بیان کنیم:
  - برنامه محاسبه میانگین نمرات را طوری تغییر دهید که محاسبات را برای هر تعداد نمره ای که کاربر اعلام می کند انجام دهد.
- در مثال قبلی تعداد نمرات مشخص بود (۱۰ عدد نمره)
- در مثال برنامه باید میانگین تعداد دلخواهی از نمرات را محاسبه کند.
- یک روش برای حل این مساله استفاده از یک مقدار خاص برای تشخیص انتهای ورود اطلاعات می باشد. به این مقدار خاص signal value یا dummy value یا flag value می گویند.

پیاده سازی الگوریتم - مثال دوم ...

- این روش را کنترل تکرار نامعلوم می نامند، چون قبل از اجرای تکرار تعداد تکرار مشخص نمی باشد.
- از آنجاییکه نمرات بین ۰ تا ۱۰۰ می باشند می توانیم عدد 1- را بعنوان علامتی برای پایان دهی به حلقه تکرار استفاده کنیم.
- برای مثال : ورودی ها به این شکل هستند

  95,96,75,74,89,-1
- در اینجا از تکنیک (top-down, stepwise refinement) که برای پیاده سازی برنامه های ساختیافته مناسب می باشد استفاده می کنیم.

پیاده سازی الگوریتم - مثال دوم ...

- ابتدا با شبه کدی شروع می کنیم که top را تعریف می کند
  - میانگین نمرات کلاس را محاسبه کن.
- در گام اول پالایش (بازنویسی) این شبه کد به صورت زیر دوباره نوشته می شود.
  - مقدار دهی اولیه به متغیرها
  - نمرات را از ورودی بخوان، جمع بزن و تعداد آنها را بشمار
  - میانگین نمرات کلاس را محاسبه و چاپ کن
- برای مرحله بعد پالایش این چنین عمل می شود

پیاده سازی الگوریتم - مثال دوم ...

- شبه کد
  - مقدار دهی اولیه به متغیرها
- بدین شکل می نویسیم
  - متغیر total را برابر صفر قرار بده
  - متغیر شمارنده counter را برابر صفر قرار بده
- شبه کد
  - نمرات را از ورودی بخوان، جمع بزن و تعداد آنها را بشمار
- بدین شکل باز نویسی می شود
  - اولین نمره را بخوان
  - تا زمانیکه کاربر عدد علامت (-1) را وارد نکرده است
    - این نمره را به مجموع نمرات اضافه کن
    - یک واحد به مقدار شمارنده اضافه کن
    - نمره بعدی را بخوان
- توجه کنید که در شبه کد از علائم آکولاد برای حلقه استفاد نکردیم و تنها دستوراتی که در حلقه تکرار اجرا می شود را با فاصله جلوتر می نویسیم.

پیاده سازی الگوریتم - مثال دوم ...

- شبه کد
  - میانگین نمرات کلاس را محاسبه و چاپ کن
- بدین شکل بازنویسی می شود
  - اگر شمارنده مساوی صفر نبود
    - میانگین را برابر تقسیم مجموع نمرات بر شمارنده قرار بده
  - در غیر اینصورت
    - عبارت اینکه "هیچ عددی وارد نشده" را چاپ کن
- در اینجا با بررسی مقدار صفربودن شمارنده از رخداد fatal error جلوگیری می کنیم
- در نهایت شبه کد ما بدین شکل می شود.

پیاده سازی الگوریتم - مثال دوم ...

- متغیر total را برابر صفر قرار بده
- متغیر شمارنده counter را برابر صفر قرار بده
- اولین نمره را بخوان
  - تا زمانیکه کاربر عدد علامت (-1) را وارد نکرده است
  - این نمره را به مجموع نمرات اضافه کن
  - یک واحد به مقدار شمارنده اضافه کن
  - نمره بعدی را بخوان
- اگر شمارنده مساوی صفر نبود
  - میانگین را برابر تقسیم مجموع نمرات بر شمارنده قرار بده
- در غیر اینصورت
  - عبارت اینکه "هیچ عددی وارد نشده" را چاپ کن

پیاده سازی الگوریتم - مثال دوم ...

- در این مثال از دو مرحله برای بازنویسی شبه کد استفاده شد. در بعضی موارد چندین مرحله نیاز می شود.
- از آنجاییکه میانگین نمرات صحیح ممکن است عددی غیر صحیح شود یعنی بخش اعشاری داشته باشد. برای محاسبه میانگین از متغیر نوع `int` نمی تواند استفاده شود.
- در این برنامه از نوع داده `float` استفاده می شود. این نوع داده برای نمایش و استفاده از داده های اعشاری استفاده می شود.
- همچنین در اینجا از عملیات `casting` برای تبدیل داده `int` به نوع `float` استفاده می شود.

پیاده سازی الگوریتم - مثال دوم ...

```
1  /* Fig. 3.8: fig03_08.c
2     Class average program with sentinel-controlled repetition */
3  #include <stdio.h>
4
5  /* function main begins program execution */
6  int main( void )
7  {
8     int counter; /* number of grades entered */
9     int grade; /* grade value */
10    int total; /* sum of grades */
11
12    float average; /* number with decimal point for average */
13
14    /* initialization phase */
15    total = 0; /* initialize total */
16    counter = 0; /* initialize loop counter */
17
18    /* processing phase */
19    /* get first grade from user */
20    printf( "Enter grade, -1 to end: " ); /* prompt for input */
21    scanf( "%d", &grade ); /* read grade from user */
22
```

Fig. 3.8 | C program and sample execution for the class average problem with sentinel-controlled repetition. (Part 1 of 3.)

پیاده سازی الگوریتم - مثال دوم ...

```
23      /* loop while sentinel value not yet read from user */
24      while ( grade != -1 ) {
25         total = total + grade; /* add grade to total */
26         counter = counter + 1; /* increment counter */
27
28         /* get next grade from user */
29         printf( "Enter grade, -1 to end: " ); /* prompt for input */
30         scanf("%d", &grade); /* read next grade */
31      } /* end while */
32
33      /* termination phase */
34      /* if user entered at least one grade */
35      if ( counter != 0 ) {
36
37         /* calculate average of all grades entered */
38         average = ( float ) total / counter; /* avoid truncation */
39
40         /* display average with two digits of precision */
41         printf( "Class average is %.2f\n", average );
42      } /* end if */
43      else { /* if no grades were entered, output message */
44         printf( "No grades were entered\n" );
45      } /* end else */
```

Fig. 3.8 | C program and sample execution for the class average problem with sentinel-controlled repetition. (Part 2 of 3.)

پیاده سازی الگوریتم - مثال دوم ...

```
46
47      return 0; /* indicate program ended successfully */
48   } /* end function main */
```

```
Enter grade, -1 to end: 75
Enter grade, -1 to end: 94
Enter grade, -1 to end: 97
Enter grade, -1 to end: 88
Enter grade, -1 to end: 70
Enter grade, -1 to end: 64
Enter grade, -1 to end: 83
Enter grade, -1 to end: 89
Enter grade, -1 to end: -1
Class average is 82.50
```

```
Enter grade, -1 to end: -1
No grades were entered
```

Fig. 3.8 | C program and sample execution for the class average problem with sentinel-controlled repetition. (Part 3 of 3.)

پیاده سازی الگوریتم - مثال دوم ...

- متغیر `average` از نوع `float` تعریف شده تا بتواند مقدار اعشاری میانگین نمرات را در برگیرد.
- از آنجاییکه در عملیات `total/counter` هر دو عملوند از نوع `int` هستند حاصل `int` خواهد بود. برای رسیدن به حاصل اعشاری با اعداد صحیح باید یک متغیر موقتی از نوع `float` بگیریم.
- زبان C عملگر قالب را برای این کار ارائه می دهد.
- خط ۳۸ :

```
average = ( float ) total / counter;
```

- عملگر قالب (cast) در این مثال یک کپی از نوع اعشاری از متغیر `total` ایجاد می کند.
- مقداری که در `total` هست همچنان عدد صحیح است.
- استفاده از عملگر قالب بدین شکل را تبدیل صریح می گویند. Explicit conversion
- با استفاده از دو پرانتز و نوع متغیر می توان عملیات تبدیل به نوع داده های دیگر را انجام داد
- عملگر قالب (cast) عملگر تکی است یعنی تنها یک عملوند دارد.

پیاده سازی الگوریتم - مثال دوم ...

- در خط ۴۱ از %.2f در تابع printf استفاده شده تا بتوان متغیر اعشاری را چاپ کرد.
- f یعنی متغیر اعشاری می خواهد چاپ شود.
- .2 دقت اعشاری را که چاپ می شود تعیین می کند. یعنی تا ۲ رقم اعشار چاپ شود.
- وقتی از %f استفاده شود به طور پیش فرض تا ۶ رقم اعشار چاپ می شود.
- استفاده از تعداد اعشار در تابع scanf اشتباه می باشد.
- وقتی عدد اعشاری با دقت مشخصی چاپ می شود، آن عدد به دقت مورد نظر گرد می شود 13.456 = 13.46 ولی مقدار واقعی متغیر در حافظه تغییر نمی کند.

```
printf( "%.2f\n", 3.446 ); /* prints 3.45 */
printf( "%.1f\n", 3.446 ); /* prints 3.4  */
```

پیاده سازی الگوریتم - مثال سوم (ساختارهای کنترل تودرتو)

- مساله :
  - آموزشگاهی درسی را ارائه می دهد که در آن دانشجویان برای اخذ مجوز دفتر ثبت آماده می شوند. سال گذشته تعدادی از دانشجویان در آزمون این درس شرکت نموده اند. آموزشگاه می خواهد بداند دانشجویانش تا چه حد در آزمون مذکور موفق بوده اند. برنامه ای بنویسید که نتایج آزمون را خلاصه کند. فهرستی شامل ۱۰ دانشجو داده میشود. در مقابل نام هر دانشجو در صورت قبولی عدد ۱ و در صورت مردودی عدد ۲ چاپ کند.

پیاده سازی الگوریتم - مثال سوم ...

- برنامه ای که نوشته می شود نتایج آزمون را بدین شکل تحلیل می کند:
  - نتیجه هر آزمون را از ورودی میخواند (۱ در صورت قبول و ۲ مردود).
    پیغام "Enter result" را برای دریافت هر یک از ورودی ها چاپ کند.
  - تعداد هر یک از نتایج آزمون را بشمارد.
  - خلاصه ای از وضعیت نتایج آزمون را چاپ کند. یعنی بگوید چند نفر قبول و چند نفر مردود شده اند.
  - اگر بیشتر از ۸ دانشجو در آزمون موفق شدند، پیغام "Bonus to Instructor" را چاپ کند.

پیاده سازی الگوریتم - مثال سوم ...

- بعد از خواندن مساله مشاهدات زیر قابل درک است:
  - برنامه باید ۱۰ نتیجه آزمون را پردازش کند. یعنی شمارنده ما باید ۱۰ بار بشمارد.
  - نتیجه هر آزمون مقدار ۱ یا ۲ دارد. برنامه باید تشخیص دهد عدد ۱ را از ورودی گرفته یا عدد ۲. در اینجا هرگاه عدد ۱ وارد شد آن را قبول و در غیر اینصورت مردود در نظر میگیریم.
  - از دو شمارنده استفاده می شود. یکی برای شمارش افراد با نمره قبول و دیگری برای شمارش افراد با نمره مردود.
  - بعد از اینکه برنامه تمام نتایج را پردازش کرد. باید تصمیم گیری کند که آیا بیشتر از ۸ نفر نمره قبولی گرفته اند یا نه.

پیاده سازی الگوریتم - مثال سوم ...

- شبه کد
  - تعداد نمرات قبول را برابر صفر قرار بده
  - تعداد نمرات مردود را برابر صفر قرار بده
  - تعداد دانشجویان را برابر ۱ قرار بده
  - تا زمانیکه تعداد دانشجویان کمتر یا مساوی ۱۰ است
    - نمره بعدی را دریافت کن
    - اگر نمره قبولی بود
      - یک واحد به تعدا قبولی ها اضافه کن
    - در غیر اینصورت
      - یک واحد به تعداد مردودی ها اضافه کن
  - تعداد قبولی ها را چاپ کن
  - تعداد مردودی ها را چاپ کن
  - اگر بیشتر از ۸ دانشجو قبول شدند
    - پیغام "Bonus to Instructor" را چاپ کن.

پیاده سازی الگوریتم - مثال سوم ...

```
1  /* Fig. 3.10: fig03_10.c
2     Analysis of examination results */
3  #include <stdio.h>
4
5  /* function main begins program execution */
6  int main( void )
7  {
8     /* initialize variables in definitions */
9     int passes = 0; /* number of passes */
10    int failures = 0; /* number of failures */
11    int student = 1; /* student counter */
12    int result; /* one exam result */
13
14    /* process 10 students using counter-controlled loop */
15    while ( student <= 10 ) {
16
17       /* prompt user for input and obtain value from user */
18       printf( "Enter result ( 1=pass,2=fail ): " );
19       scanf( "%d", &result );
20
21       /* if result 1, increment passes */
22       if ( result == 1 ) {
23          passes = passes + 1;
```

پیاده سازی الگوریتم - مثال سوم ...

```
24          } /* end if */
25          else {/* otherwise, increment failures */
26             failures = failures + 1;
27          } /* end else */
28
29          student = student + 1; /* increment student counter */
30       } /* end while */
31
32       /* termination phase; display number of passes and failures */
33       printf( "Passed %d\n", passes );
34       printf( "Failed %d\n", failures );
35
36       /* if more than eight students passed, print "Bonus to instructor!" */
37       if ( passes > 8 ) {
38          printf( "Bonus to instructor!\n" );
39       } /* end if */
40
41       return 0; /* indicate program ended successfully */
42    } /* end function main */
```

پیاده سازی الگوریتم - مثال سوم ...

```
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 2
Enter Result (1=pass,2=fail): 2
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 2
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 2
Passed 6
Failed 4
```

Fig. 3.10 | C program and sample executions for examination results problem. (Part 3 of 4.)

پیاده سازی الگوریتم - مثال سوم ...

```
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 2
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Passed 9
Failed 1
Bonus to instructor!
```

Fig. 3.10 | C program and sample executions for examination results problem. (Part 4 of 4.)

عملگرهای انتساب

- در زبان C عملگرهای خلاصه بسیاری برای عملگرهای انتساب ارائه شده

```
c = c + 3;          c += 3;
```

- به طور کلی تمام دستورات به شکل

```
variable = variable operator expression;
```

- که در آن operator یکی از عملگرهای +-*/% است می تواند به این شکل نوشته شود.

```
variable operator=  expression;
```

عملگرهای افزایشی و کاهشی

- زبان C همچنین عملگرهای افزایشی و کاهشی ++ و -- را ارائه میدهد. این عملگرها تک عملوند هستند.
- x = x +1 و x += 1 و x++ با هم معادلند
- x = x -1 و x-= 1 و x– با هم معادلند
- دو دستور x++ و ++x با هم متفاوتند
- تفاوت این دو دستور در تقدم و تاخر عملیات افزایش است هنگامیکه از این دستور در میان دستورات دیگر استفاده می کنیم.
- X++ ابتدا از متغیر x را استفاده می کند سپس به آن اضافه می کند.
- ++x ابتدا یک واحد به x اضافه می کند سپس از آن استفاده می کند.

عملگرهای افزایشی و کاهشی

```
1  /* Fig. 3.13: fig03_13.c
2     Preincrementing and postincrementing */
3  #include <stdio.h>
4
5  /* function main begins program execution */
6  int main( void )
7  {
8     int c; /* define variable */
9
10    /* demonstrate postincrement */
11    c = 5; /* assign 5 to c */
12    printf( "%d\n", c ); /* print 5 */
13    printf( "%d\n", c++ ); /* print 5 then postincrement */
14    printf( "%d\n\n", c ); /* print 6 */
15
16    /* demonstrate preincrement */
17    c = 5; /* assign 5 to c */
18    printf( "%d\n", c ); /* print 5 */
19    printf( "%d\n", ++c ); /* preincrement then print 6 */
20    printf( "%d\n", c ); /* print 6 */
21    return 0; /* indicate program ended successfully */
22 } /* end function main */
```

عملگرهای افزایشی و کاهشی

```
5
5
6
 5
5
6
6
```

Fig. 3.13 | Preincrementing vs. postincrementing. (Part 2 of 2.)

سوال

- خروجی این برنامه چیست اگر k=1 باشد

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

- اگر خط ۲ بدین شکل باشد

```
i -= --i;
```

- خروجی چه می شود.

سوال

- اگر i=0

```
1. printf("%d", ++(++i));
```

- 2

```
2. printf("%d", ++++i);
```

- 2

```
3. printf("%d", ++i++);
```

- Error

```
4. printf("%d", (++i)++);
```

- 1

```
3. printf("%d", ++(i++));
```

- Error