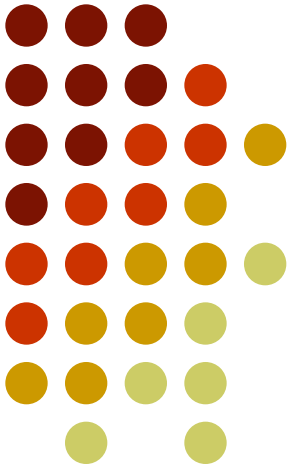
مبانی کامپیوتر و برنامه سازی

فصل هشتم : ساختارهاي كنترلي

مدرس : سعید ابریشمی



8 ساختارهاي كنترلي

- ساختارها در برنامه نویسی ساختیافته
  - ساختار ترتیب
  - ساختار انتخاب
  - ساختار تکرار
- زبان C دارای 7 نوع ساختار کنترلي است
- ساختار ترتیب: دستورهای زبان C در حالت عادی به همان ترتیبی که نوشته شده اند، یکی پس از دیگری اجرا می شوند.
- 3 نوع ساختار انتخاب:
  - ساختار if یا ساختار تک انتخابی
  - ساختار if / else یا ساختار دو انتخابی
  - ساختار switch یا ساختار چند انتخابی
- 3 نوع ساختار تکرار
  - while
  - for
  - do / while

1-8 ساختار انتخاب if

- این دستور به شکل زیر استفاده می شود:

```
if (<expresion>) <statement>;
```

- نحوه کار بدینصورت است که ابتدا عبارت موجود در قسمت <expression> ارزیابی می شود. در صورتیکه درست ارزیابی گردد، دستور قسمت <statement> اجرا خواهد شد و در صورتیکه نادرست باشد، بدون اینکه دستور قسمت <statement> را اجرا کند به دستور بعدی خواهد رفت.

- این دستور می تواند بصورت زیر نیز استفاده گردد:

```
if (<expresion>) <statement 1>;
else <statement 2>;
```

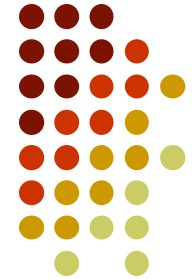
- در اینصورت ابتدا عبارت موجود در قسمت <expression> ارزیابی می شود. در صورتیکه درست ارزیابی گردد، دستور قسمت <statement 1> اجرا خواهد شد، و در صورتیکه نادرست باشد، دستور قسمت <statement 2> اجرا خواهد شد. در هر حال فقط یکی از این دو قسمت اجرا خواهد گردید.

1-8 ساختار انتخاب if

- بعنوان مثال چنانچه متغیر grade حاوی نمره دانشجو باشد و بخواهیم بر مبنای نمره وی، پیغام مناسبی چاپ کنیم، می توانیم از دستور زیر استفاده کنیم:

```
if  (grade >= 10)  printf("Passed !");
else  printf("Failed!");
```

- در حالت عادی دستور if منتظر یک دستور در بدنه خود می باشد، اما چنانچه می خواهید چندین دستور را در بدنه یک دستور if دهید، باید آنها را در داخل آکولاد باز وبسته { } قرار دهید. این مجموعه دستورات را یک **دستور مرکب** می گویند.
- بطور کلی در زبان C هرجا که می توان یک دستور قرار داد، می توان از یک دستور مرکب نیز استفاده کرد. به یک دستور مرکب، **بلوک** نیز گفته می شود.



1-8 ساختار انتخاب if

- بنابراین صورت کلی دستور if به شکل زیر است:

```
if  (<expression>) {
      <statement 1> ;
      <statement 2> ;
       ….
      <statement n> ;
}
else {
      <statement 1> ;
      <statement 2> ;
       …….
      <statement m> ;
}
```

1-8 ساختار انتخاب if

- برنامه 1) برنامه ای بنویسید که ضرایب یک معادله درجه 2 را دریافت و ریشه های آن را محاسبه و چاپ نماید.

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

void main() {
   int  a, b, c;
   float  x1, x2, delta;
   clrscr();
   printf("Please enter a, b and c : ");
   scanf("%d %d %d", &a, &b, &c);
   if (a==0) {
       printf("wrong equation!");
       exit(1) ;
   }
```

if (!a)

1-8 ساختار انتخاب if

```
delta = b*b – 4*a*c;
if  (delta < 0)
    printf(“No answer !”);
else if (delta == 0) {
        x1 = -b / (2*a);
        printf(“There is one answer, x = %8.2f”,x1);
      }
      else {
        delta = sqrt(delta);
        x1 = (-b+delta) / (2*a);
        x2 = (-b-delta) / (2*a);
        printf(“There are two answers, x1= %8.2f and x2 = %8.2f”, x1, x2);
      }
}
```

1-8 ساختار انتخاب if

- يك روش متداول استفاده از دستور if، استفاده از if هاي تودرتو مي باشد.

```
if  (grade >= 18)  printf("good!");
else if (grade >= 15)  printf("medium!");
      else if  (grade >= 12) printf("rather weak!");
            else if (grade >= 10) printf("weak");
                  else  printf("failed!");
```

- درچنين حالتي توصيه مي گردد كه شرطهاي نادر را كه امكان وقوع آنها كم است، در انتهاي كار بررسي نماييد، تا تعداد مقايسه كمتري صورت پذيرد.

1-8 ساختار انتخاب if

- مشكل if هاي تودرتو : در دستور زير، else به كدام if تعلق دارد؟

```
if  (a < b)
    if (c < d)  <statement 1>;
    else <statement 2>;
```

- بطور كلي طبق قوانين گرامري، هر else مربوط به نزديكترين if قبل از خود مي باشد.

1-8 ساختار انتخاب if

- اما سوال این است که اگر بخواهیم else به if اول بازگردد از چه روشي استفاده نماییم. دراینصورت مي توان از يكي از دو روش زیر استفاده کرد:

```
if  (a < b)
   if (c < d)  <statement 1>;
   else ;
else <statement 2>;
```

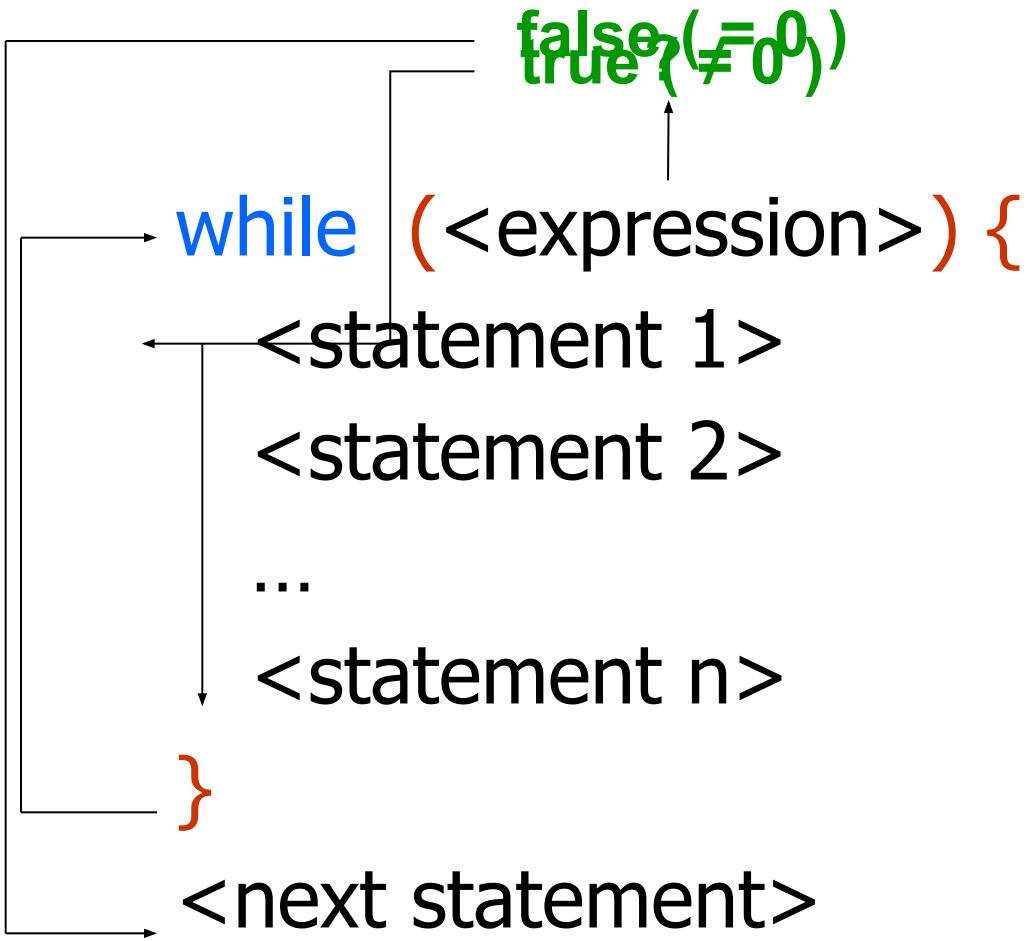
```
if  (a < b) {
    if (c < d)  <statement 1>;
}
else <statement 2>;
```

1-8 ساختار انتخاب if

- برنامه 2) برنامه اي بنويسيد كه 3 عدد را دريافت و حداكثر آنها را چاپ كند.

```
#include <stdio.h>
void main() {
   int  a, b, c, max;

   printf("Please enter 3 numbers :");
   scanf("%d %d %d",&a, &b, &c);

   if (a > b)
      if (a > c)  max = a;
      else  max= c;
   else  if (b > c)  max = b;
         else  max = c;
   printf("Maximum is %d",max);
}
```

2-8 ساختار تکرار while



2-8 ساختار تکرار while

- برنامه 3) برنامه اي بنويسيد كه يك عدد را دريافت و فاكتوريال آن را محاسبه و چاپ نمايد.

```
#include <stdio.h>
void main() {
   int  i,number;
   long int  factorial;
   printf("Please enter number :");
   scanf("%d",&number);

   factorial = 1;
   i = 1;
   while (i <= number) {
       factorial *= i;
       i ++;
   }
   printf("Factorial of %d is %ld‟,number,factorial);
}
```

2-8 ساختار تکرار while

- برنامه 4) برنامه اي بنويسيد كه يك متن را از كاربر دريافت و آن را با حروف بزرگ چاپ كند.

```
#include <conio.h>

void main() {
   char  ch;

   ch = getch() ;
   while (ch != 13) {
       if (ch >= 'a'  && ch <= 'z')
           ch -= 32;
       putch(ch);
       ch = getch();
   }
}
```

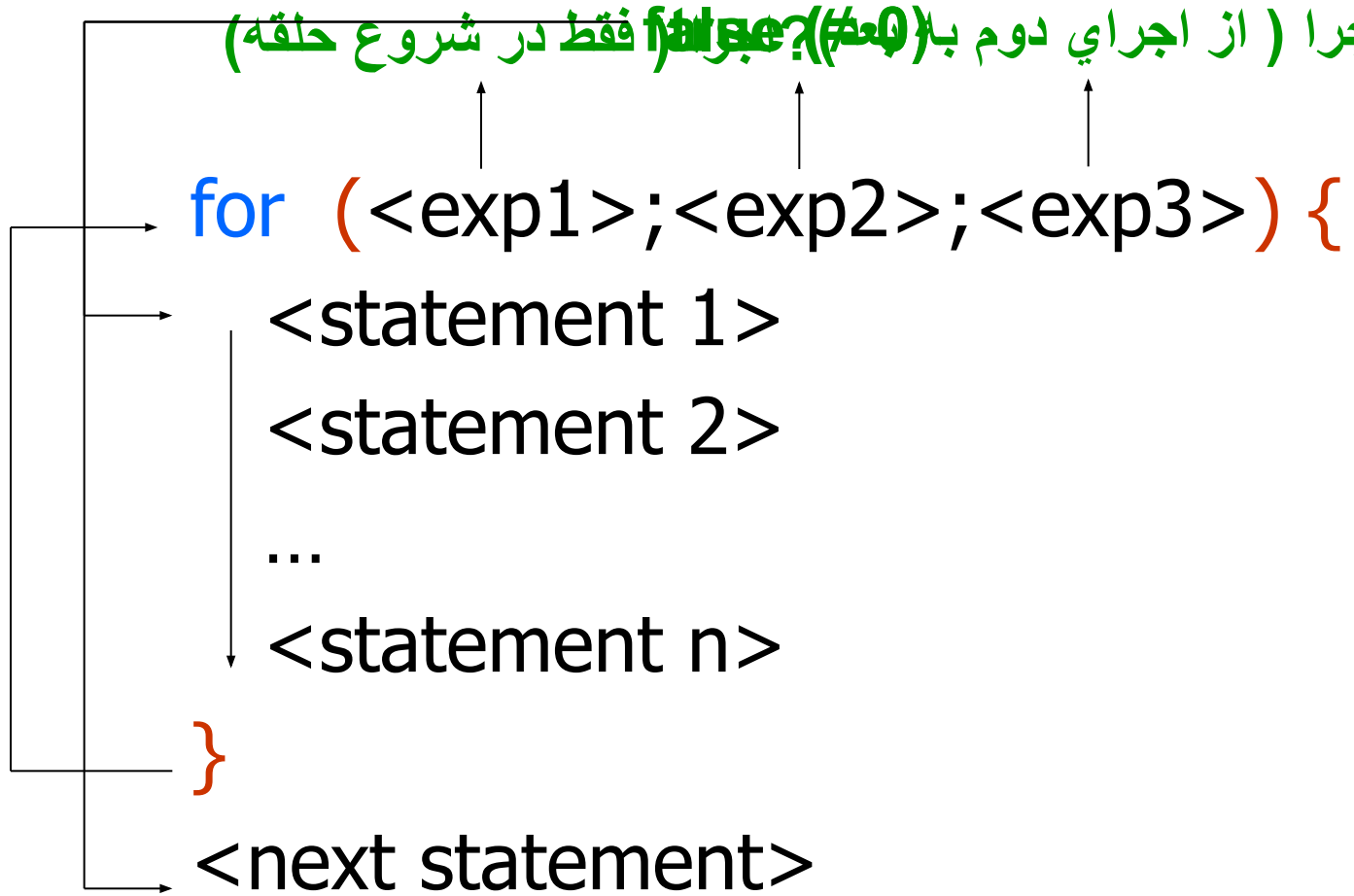
3-8 ساختار تکرار for

- همانگونه که در مثال مربوط به حل مسئله فاکتوریال دیده مي شود، گاهي نياز به حلقه تکراري داريم که به تعداد دفعات مشخصي تکرار گردد.
- در چنين مواقعي با استفاده از يك متغير شمارنده، تعداد تکرارها را تا رسیدن به مقدار مورد نظر می شماریم و سپس به حلقه پایان می دهیم. به چنین حلقه هایی، **تکرار تحت کنترل شمارنده** یا **تکرار معین** می گوییم، چرا که تعداد تکرار ها از قبل مشخص است.
- چنین حلقه ای دارای 3 جزء اصلی می باشد:
  - مقداردهی اولیه به متغیر شمارنده حلقه
  - شرط پایان حلقه (پایان شمارش)
  - نحوه افزایش متغیر شمارنده
- از آنجا که در تمام حلقه هایی که تکرار معین دارند، همین ساختار استفاده می شود؛ در اکثر زبانهای برنامه سازی یک ساختار تکرار ویژه، بنام حلقه for، برای اینکار در نظر گرفته شده است.

3-8 ساختار تکرار for

```
for (<expression1> ; <expression2> ; <expression3>) <statement>;
```

3-8 ساختار تکرار for



3-8 ساختار تکرار for

- درحقیقت هر حلقه for معادل با حلقه while زیر است:

```
<exp1> ;
while (<exp2>) {
   <statement>;
   <exp3>;
}
```

3-8 ساختار تکرار for

- بعنوان یک مثال ساده، تکه برنامه زیر اعداد بین 0 تا 100 را چاپ می نماید:

```
int  count;
for  (count = 0; count <= 100; count ++)
   printf("%d ",count);
```

- اگر بخواهیم تنها مضارب 5 چاپ شوند، حلقه را به شکل زیر تغییر مي دهيم:

```
for  (count = 0; count <= 100; count += 5)
```

- حتي مي توان مضارب 5 را از آخر به اول چاپ کرد:

```
for  (count = 100; count >= 0; count -= 5)
```

- قسمت شرط مي تواند يك شرط مركب نيز باشد.

```
for  (count = 0; count < 100 && sw==1; count ++)
```

- نكته آخر اينكه قسمت مقدار دهي اوليه و افزايش متغير نيز مي توانند شامل چند عبارت باشند كه در اينصورت با كاما از يكديگر جدا مي شوند.

```
for  (a = 0, b = 100; b – a > 50;  a++, b--)
```

3-8 ساختار تکرار for

- برنامه 5) برنامه ای بنویسید که تعدادی عدد را از کاربر دریافت و 2 عدد بزرگتر و مجموع کل اعداد را محاسبه و چاپ نماید.

```
#include <stdio.h>
void main() {
   int  i, n, number;
   int  sum, max1, max2;
   printf("please enter n : ");
   scanf("%d",&n);

   sum = 0;
   max1 = max2 = -1;
   for (i=0 ; i<n ; i++) {
        printf("enter number : ");
        scanf("%d",&number);

        sum += number;
        if (number > max1) {
             max2 = max1;
             max1 = number;
        }
        else if (number > max2)
                 max2 = number;
   }  //end for
   printf("Sum = %d, Maximum 1=%d, Maximum 2= %d", sum, max1, max2);
}
```

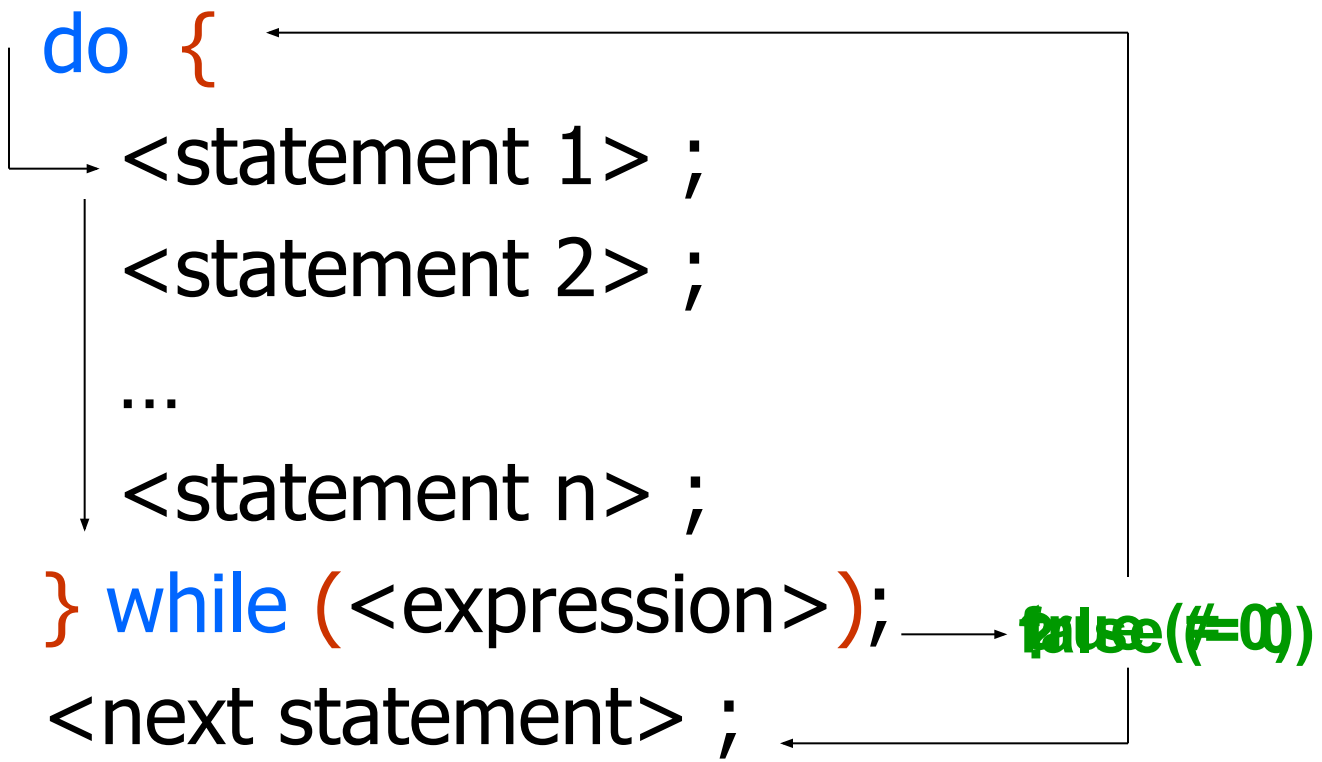
3-8 ساختار تكرار for

- نكته جالب در مورد حلقه for آنستكه مي توان هريك از 3 عبارت آن را حذف كرد.

```
for (; i<100; i++)

for  (i=0; i<100;)

for  (; i<100;)

for (i=0; ;i++)
```

- در مورد آخر حتما بايد در داخل حلقه با استفاده از دستور break (كه در قسمتهاي بعدي توضيح داده خواهد شد)، راهي براي خروج از حلقه قرار داده شود.

4-8 حلقه do / while



4-8 حلقه do / while

- يك مثال كوچك:
- فرض كنيد از كاربر خواسته ايد كه اعلام كند آيا مايل به ادامه هست يا خير؟ وي بايد پاسخ y يا n بدهد، اما ممكن است يك حرف اشتباه (مانند m) وارد كند.
- قصد داريم تكه برنامه اي بنويسيم كه عمل دريافت پاسخ را تا زمانيكه يك حرف درست وارد شود، تكرار كند.
- مسلم است كه بايد ابتدا يك پاسخ وارد شود و سپس درستي آن بررسي گردد.

```
char  answer;
do {
   printf("Do you want to continue (y/n) ?");
   answer = getch();
} while (answer != 'y' && answer != 'n') ;
```

4-8 حلقه do / while

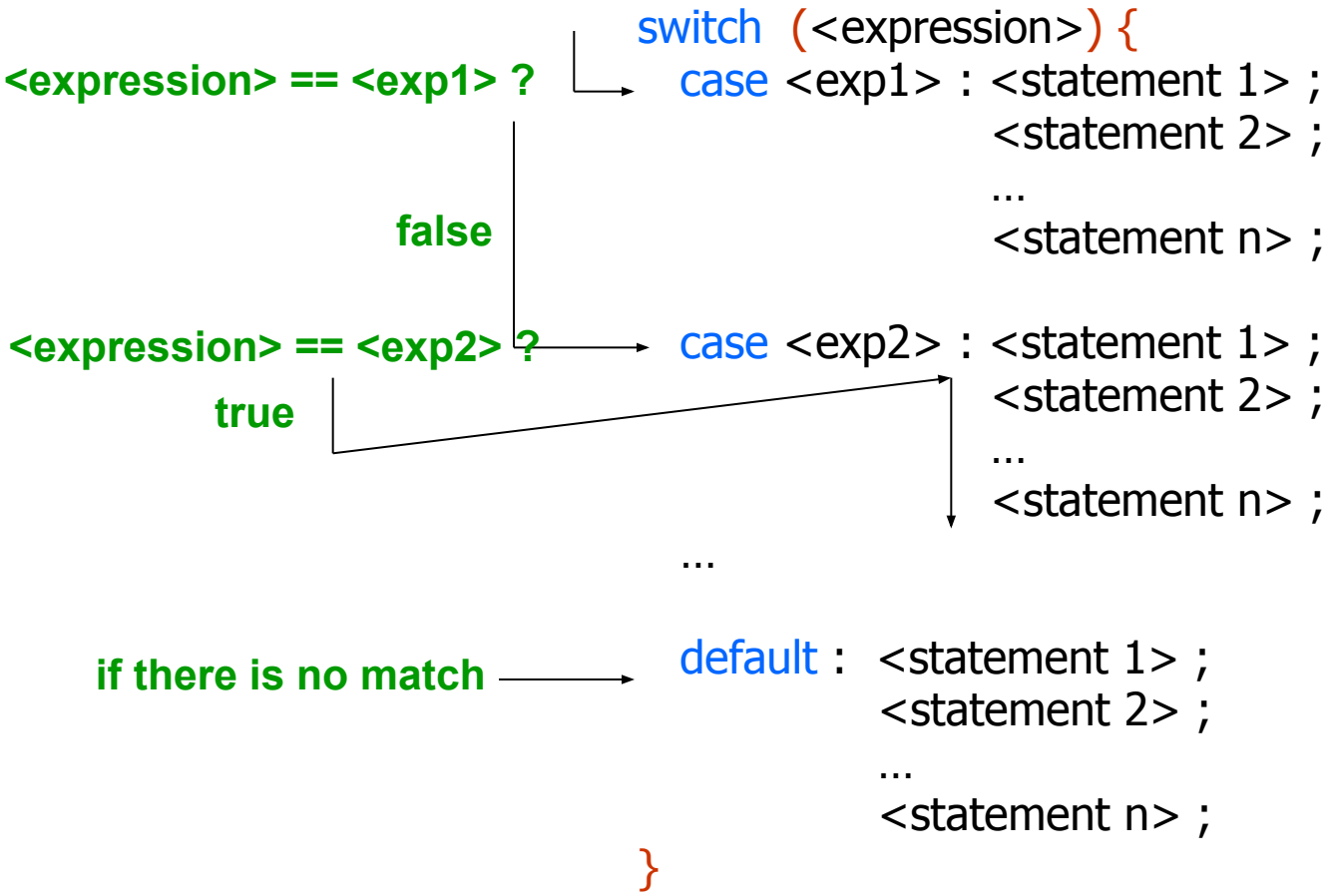
- برنامه 6) فرض كنيد نمرات يك گروه از دانشجويان بصورت درجه بندي (A, B, C and D) آماده شده است. برنامه اي بنويسيد كه نمرات دانشجويان را دريافت و در پايان درصد هريك از نمرات را محاسبه و چاپ نمايد. در ضمن از آنجا كه تعداد دانشجويان از قبل مشخص نيست، كاربر در انتهاي نمرات، حرف Q (مخفف Quit) را وارد مي نمايد.

```
#include <stdio.h>
void main() {
   int  aCount, bCount, cCount, dCount, n;
   char  grade;
   aCount = bCount = cCount = dCount = n = 0;
   do {
        printf("Enter grade (Q for Quit) : ");
        grade = getch() ;

        n ++;
        if (grade == 'A') aCount ++;
        else if (grade == 'B')  bCount ++;
             else if (grade == 'C')  cCount ++;
                  else if (grade == 'D')  dCount ++;
                       else if (grade == 'Q')  n --;
                            else  {
                                 printf("Wrong grade, try again.\n");
                                 n --;
                             }
   } while (grade != 'Q' ) ;
```

4-8 حلقه do / while

```
printf("Statistics :\n");
printf("Grade A : %f percent\n", float(aCount)/float(n));
printf("Grade B : %f percent\n", float(bCount)/float(n));
printf("Grade C : %f percent\n", float(cCount)/float(n));
printf("Grade D : %f percent\n", float(dCount)/float(n));

} // end main
```

5-8 ساختار switch / case



5-8 ساختار switch / case

- توجه کنید که قسمت default اختیاری بوده و می توان از آن استفاده نکرد.
- این ساختار فقط برای عبارات کاراکتری و صحیح معتبر بوده و نمی توان در آن از عبارات اعشاری استفاده نمود.
- نکته مهم دیگر در مورد این ساختار این است که چنانچه عبارت <expression> با یک ثابت مانند <constant i> برابر باشد، آنگاه پس از اینکه دستورات مربوط به این حالت اجرا گردید، اجرا ادامه یافته و دستورات مربوط حالتهای بعدی تا انتهای switch انجام خواهد شد!
  - به عنوان مثال چنانچه عبارت <expression> با ثابت <constant 2> برابر باشد، پس از اجرای دستورات مربوط به این حالت، دستورات حالتهای <constant 3> و ... تا <constant m> و حتی قسمت default نیز اجرا خواهد گردید.
- برای جلوگیری از این وضعیت که معمولا دلخواه برنامه نویسان نیست، می توان از دستور break استفاده کرد. این دستور که بعدا در مورد آن توضیح بیشتری خواهیم داد، باعث می شود که از ساختار switch خارج شده و به دستور پس از آن برویم.
- بنابراین معمولا برنامه نویسان در پایان دستورات هر case، از یک دستور break استفاده می کنند. این کار باعث می شود که پس از اجرای دستورات مربوط به هر case، با رسیدن به دستور break بلافاصله از ساختار switch خارج شده و دستورات مربوط به case بعدی اجرا نشوند.

5-8 ساختار switch / case

- اما چرا در زبان C از این روش استفاده شده است بطوریکه برنامه نویسان مجبور به استفاده از دستور break شوند؟
  - جواب این است که می توان با استفاده از این خاصیت، چندین case مختلف را با یکدیگر یای منطقی (or) کرد.
  - فرض کنید چند case مختلف دارید که قصد دارید با وقوع هریک از آنها، مجموعه دستورات مشترکی انجام شوند.
  - کافی است این case ها را بصورت پشت سرهم قرار داده و دستورات همگی آنها بجز case آخر را خالی قرار دهید.
  - حال دستورات مشترک را در case آخر قرار داده و در انتها نیز یک دستور break بگذارید.
  - اکنون چنانچه عبارت با هریک از این case ها برابر باشد، از آنجا که هیچیک دارای دستور break نیستند، اجرا تا case آخر ادامه خواهد یافت و در پایان دستورات مشترک اجرا خواهد شد.

5-8 ساختار switch / case

- بعنوان مثال فرض کنید یک متغیر صحیح بنام point داریم که امتیاز یک ورزشکار را بین 1 تا 5 مشخص می نماید. اکنون قصد داریم بسته به امتیاز ورزشکار، پیام مناسبی را برای وی چاپ نماییم. امتیاز 1 یا 2 ضعیف، امتیاز 3 متوسط، و امتیاز 4 یا 5 خوب ارزیابی می گردد. ساختار زیر این کار را انجام می دهد.

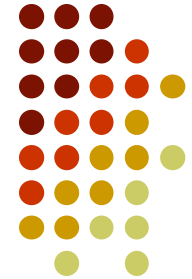
```
switch (point) {
    case 1 :
    case 2 : printf("weak!\n");
             break;
    case 3 : printf("medium!\n");
             break;
    case 4 :
    case 5 : printf("good!\n");
             break;
    default : printf("out of range!");
}
```

5-8 ساختار switch / case

- برنامه 7) برنامه 6 را با استفاده از دستور switch / case بازنويسي نماييد. برنامه را بگونه اي بنويسيد كه حروف بزرگ و كوچك هردو مورد قبول واقع شود.

```
#include <stdio.h>
void main() {
   int  aCount, bCount, cCount, dCount, n;
   char  grade;
   aCount = bCount = cCount = dCount = n = 0;
   do {
        printf("Enter grade (Q for Quit) : ");
        grade = getch() ;

        n ++;
```

5-8 ساختار switch / case



```
        switch (grade) {
          case 'A' :
          case 'a' : aCount ++;      break ;
          case 'B' :
          case 'b' : bCount ++;      break ;
          case 'C' :
          case 'c' : cCount ++;      break ;
          case 'D' :
          case 'd' : dCount ++;      break ;
          case 'Q' :
          case 'q'  :  n--;          break ;
          default  :  printf("Wrong grade, try again.\n");
                      n --;
        } //end switch

    } while (grade != 'Q' && grade!='q') ;

    printf("Statistics :\n");
    printf("Grade A : %f percent\n", float(aCount)/float(n));
    printf("Grade B : %f percent\n", float(bCount)/float(n));
    printf("Grade C : %f percent\n", float(cCount)/float(n));
    printf("Grade D : %f percent\n", float(dCount)/float(n));
} // end main
```

5-8 ساختار switch / case

- برنامه 8) برنامه اي بنويسيد كه يك عدد، يك عملگر و يك عدد ديگر را از كاربر دريافت و پس از اعمال عملگر برروي دو عدد، حاصل را چاپ نمايد.

```
#include <stdio.h>
void main() {
   int  number1, number2, result;
   char  op ;
   printf("Please enter number1 operator number2 : ");
   scanf("%d %c %d",&number1, &op, &number2);

   result = 0;
   switch (op) {
       case '+' : result = number1 + number2 ;    break;
       case '-'  : result = number1 - number2 ;    break;
       case '*'  : result = number1 * number2 ;    break;
       case '/'  : if (number2 != 0) result = number1 / number2 ;
                   else   printf("There is no answer!\n");
                   break;
       case '%'  : if (number2 != 0) result = number1 % number2 ;
                     else  printf("There is no answer!\n");
                     break;
        default : printf("invalid operator!\n");
    }
     printf("Result = %d",result);
}
```

6-8 دستور break

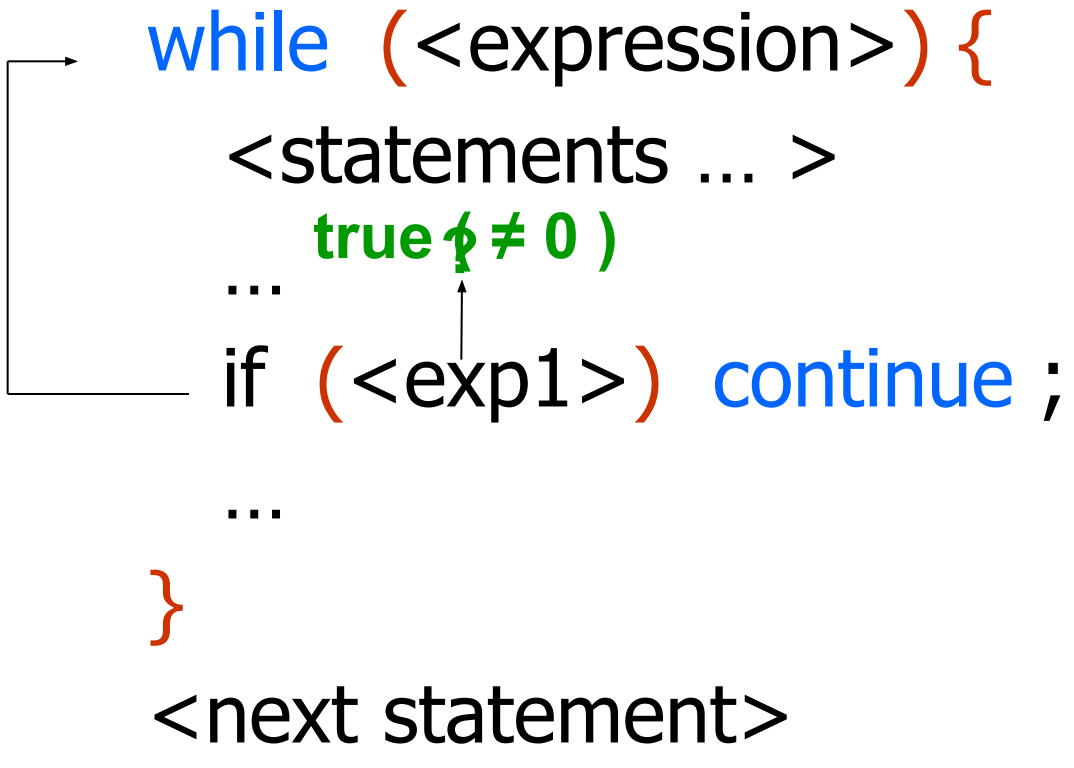
```
while (<expression>) {
    <statements ... >
        true ( ≠ 0 )
    ...
    if (<exp1>) break ;
    ...
}
<next statement>
```

6-8 دستور break

- برنامه 9) برنامه 5 را بگونه اي تغيير دهيد كه فقط اعداد مثبت را بپذيرد، و درصورتيكه عدد منفي وارد شد، بلافاصله به عمليات خاتمه داده و نتايج تا همين نقطه را چاپ نمايد.

```
#include <stdio.h>
void main() {
   int  i, n, number;
   int  sum, max1, max2;
   printf("please enter n : ");
   scanf("%d",&n);

   sum = 0;
   max1 = max2 = -1;
   for (i=0 ; i<n ; i++) {
        printf("enter number : ");
        scanf("%d",&number);
        if (number < 0) break;  // this is the difference

        sum += number;
        if (number > max1) {
             max2 = max1;
             max1 = number;
        }
        else if (number > max2)
                 max2 = number;
   }  //end for
   printf("Sum = %d, Maximum 1=%d, Maximum 2= d", sum, max1, max2);
}
```

6-8 دستور continue



6-8 دستور continue

- بعنوان مثال، چنانچه بخواهيم برنامه 9 را بگونه اي تغيير دهيم كه از اعداد منفي صرفنظر كند و آنها را در محاسبات لحاظ نكند، كافيست دستور

```
if  (number < 0) break;
```

را به دستور زير تبديل كنيم:

```
if (number < 0) continue;
```

- دراينصورت، چنانچه عدد منفي باشد، بدون اينكه محاسبات بعدي انجام شوند، كنترل به ابتداي حلقه بازگشته و عدد بعدي را دريافت مي كند.