دانشگاه بین المللی امام خمینی قزوین

دانشکدهٔ فنی و مهندسی

مبانی کامپیوتر و برنامه سازی

(دومین ترم کرونا!)

فصل ششم: ساختارهای کنترلی در زبان C

نستوه طاهری جوان

nastooh@aut.ac.ir

ساختارهای کنترلی

- یادآوری: دستورات یک برنامه به ترتیب از ابتدا اجرا می شوند.
  - تا اجرای یک دستور تکمیل نشود، دستور بعدی شروع نمی شود.

- نکته: در برنامه های واقعی گاهی نیاز است دستوراتی چندین بار تکرار شوند و یا دستوراتی فقط تحت شرایط خاصی اجرا شوند (و نه همیشه!).

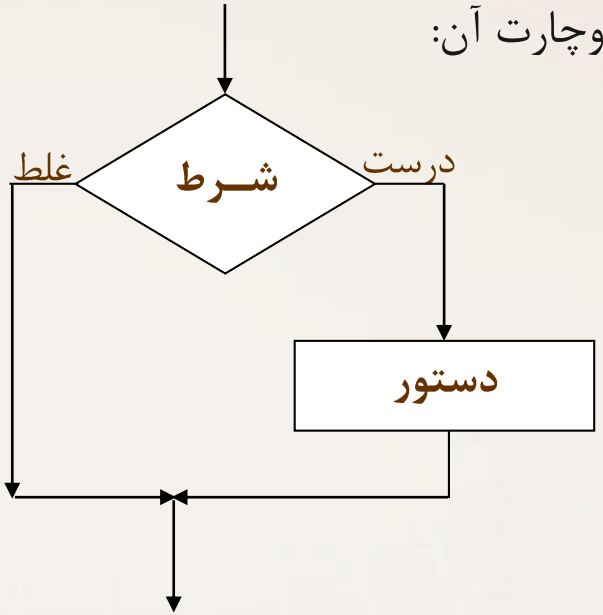
nastooh@aut.ac.ir

ساختارهای تصمیم

- دستور if
  - یک شرط را چک کرده و در صورتی که شرط برقرار باشد، دستور خاصی اجرا می شود.
  - نحوه استفاده:

```
if (شرط)
    دستور;
```

  - فلوچارت آن:



nastooh@aut.ac.ir

ساختارهای تصمیم

- دستور if
  - مثال ساده:

```
int  i;
scanf("%d", i);
if (i > 0)
    printf("positive");
```

- نکته: بعد از دستور if فقط و فقط یک دستور را می توان شامل حال شرط آن کرد.
- مثال:

```
if (i > j)
    printf("1");
printf ("2");
```

نتیجه:

عبارت ۲ همواره چاپ خواهد شد. (بدون توجه به شرط)

nastooh@aut.ac.ir

ساختارهای تصمیم

- دستور if
  - **نکته مهم:**

اگر نیاز است در صورت برقراری شرط، بیش از یک دستور شامل حال شرط شوند، باید دستورات را بین آکولاد باز و بسته (به عنوان یک بلاک!) قرار داد.

مثال:

```
if (i > 0)
{
    printf("1");
    printf("2");
    printf("3");
}
```

نتیجه:

در صورت برقراری شرط، هر سه دستور اجرا خواهند شد و در غیر این صورت هیچ یک اجرا نخواهند شد.

nastooh@aut.ac.ir

ساختارهای تصمیم

- عادت مهم در برنامه نویسی
  - ***رعایت تو رفتگی ها***

- دستورات بین دو آکولاد (اصطلاحاً یک بلاک برنامه نویسی) باید قدری تو رفته تر تایپ شوند. (مثلا به اندازهٔ ۴ یا ۵ فاصله)
- برای قرار دادن آکولاد دو روش زیر مرسوم است:

```
if (i > 0) {
    printf("1");
    if (j > 5) {
        printf("2");
        printf("3");
    }
    printf ("4");
}
```

```
if (i > 0)
{
    printf("1");
    if (j > 5)
    {
        printf("2");
        printf("3");
    }
    printf ("4");
}
```


nastooh@aut.ac.ir

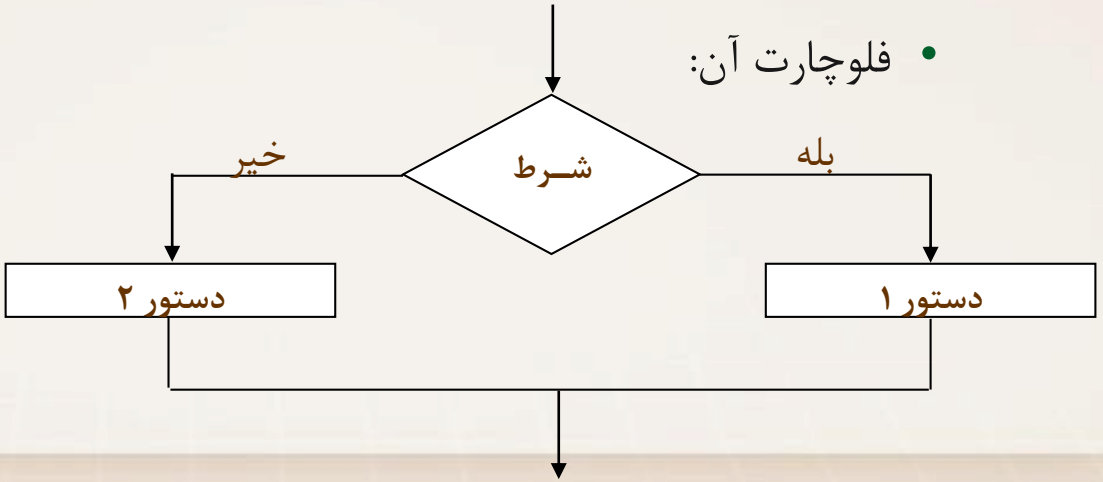
ساختارهای تصمیم

دستور if-else

- یک شرط را چک می کند، در صورت برقراری آن شرط دستور خاصی را اجرا می کند و در صورت عدم برقراری شرط، دستور خاص دیگری را اجرا می کند.
- نحوه استفاده:

```
if (شرط)
        دستور یک;
else
        دستور دو;
```

- فلوچارت آن:



nastooh@aut.ac.ir

ساختارهای تصمیم

- دستور if-else
  - مثال:

```
if (a > b)
    x = 5;
else
    x = -5;
```

- بعد از if یا بعد از else مربوط به آن فقط یک دستور را می توان قرار داد و برای قرار دادن چند دستور باید از آکولاد باز و بسته استفاده کرد.

```
if (b = = 0)
    printf(“zero”);
else
{
    printf(“XXX”);
    y = 5;
}
```


nastooh@aut.ac.ir

ساختارهای تصمیم

- مثال ساده:

- برنامه ای که یک عدد صحیح را دریافت کرده و مشخص می کند آن عدد صفر است یا خیر؟

```
#include<stdio.h>
void main()
{
    int  number;
    printf(“please enter a number:”);
    scanf(“%d”, &number);
    if (number  = = 0)
        printf(“%d  is  ZERO”);
    else
        printf(“%d  is not ZERO”);
    getchar();
}
```

نتیجه: مثلا با ورودیِ 120:

120 is not ZERO

nastooh@aut.ac.ir

ساختارهای تصمیم

- تمرین در کلاس:

  - برنامه ای بنویسید که یک عدد را دریافت کرده و قد مطلق آن را نمایش دهد.
  - برنامه ای بنویسید که دو عدد را از ورودی دریافت کرده و عدد بزرگتر را نمایش دهد.
  - برنامه ای بنویسید که یک عدد را دریافت کرده و مشخص کند آن عدد زوج است یا خیر؟

nastooh@aut.ac.ir

ساختارهای تصمیم

- if های تو در تو
  - برای حالت های پیچیده تر، می توان از if های تو در تو استفاده کرد.

- **در این حالت، هر else مربوط به نزدیک ترین ifای است که تا به حال به else دیگری نسبت داده نشده است.**
  - **برای تحلیل قطعه کدها، از داخلی ترین else شروع کنید و با قاعدهٔ فوق، elseها را به ifها منتسب کنید، تا عملکرد قطعه کد را درک کنید.**

nastooh@aut.ac.ir

ساختارهای تصمیم

- if های تو در تو

- مثال: قطعه کد زیر را تحلیل کنید!

```
int  x, y;
scanf("%d%d", &x, &y);
if (x != 0)
    if (y != 0)
        printf("case1");
    else
        printf("case2");
else
    printf("case3");
```


nastooh@aut.ac.ir

ساختارهای تصمیم

if های تو در تو

- مثال: قطعه کدهای زیر را تحلیل کنید!

➢ احتمالاً هدف ما از این قطعه کدها این بوده است که با ورود عدد ۱ عبارت one، و با ورود عدد ۲ عبارت two و با ورود عدد ۳ عبارت three و با ورود اعدادی دیگر عبارت other نمایش داده شود. اما قطعه کد سمت راست چه رفتاری دارد؟ خوب دقت کنید!

```
int i;
scanf("%d",&i);
if (i = = 1)
    printf("one");
else
    if (i = = 2)
        printf("two");
    else
        if (i = = 3)
            printf("three");
        else
            printf("other");
```

```
int i;
scanf("%d",&i);
if (i = = 1)
    printf("one");
if (i = = 2)
    printf("two");
if (i = = 3)
    printf("three");
else
    printf("other");
```


nastooh@aut.ac.ir

ساختارهای تصمیم

- تمرین در کلاس:
  - برنامه ای بنویسید که که سه عدد را دریافت کرده و به صورت زیر عمل کند:
    - اگر عدد سوم برابر صفر بود، حاصلضرب دو عدد اول را نمایش دهد
    - اگر عدد سوم منفی بود، تفاضل دو عدد اول را نمایش دهد.
    - اگر عدد سوم مثبت بود، حاصلجمع دو عدد اول را نشان دهد.

  - برنامه ای بنویسید که یک عدد ۵ رقمی را از ورودی دریافت کرده، و مشخص کند جناس قلب دارد یا خیر؟
    - اعدادی جناس قلب دارند، که از هر دوطرف یکسان خوانده شوند، مثلا ۳۶۷۶۳
    - فعلاً فقط و فقط بر روی اعداد ۵ رقمی تمرکز کنید (نه بزرگتر و نه کوچکتر!).

nastooh@aut.ac.ir

ساختارهای تصمیم

- نکته:
  - در زبان C می توان شرط مخالف صفر بودن را به صورت زیر کوته نویسی کرد. به عبارت دیگر، دو شرط زیر با هم برابرند:

```
if ( i )
```

```
if ( i != 0 )
```


nastooh@aut.ac.ir

ساختارهای تصمیم

ساختار switch

- یک ساختار خوانا برای پیاده سازی if های متوالی بر اساس مقادیر مختلف یک عبارت
- نحوه استفاده:

```
switch (متغیر یا عبارت)
{
 case  مقدار ثابت اول :
          دستورات مورد نظر;
          break;
 case  مقدار ثابت دوم :
          دستورات مورد نظر;
          break;
 case  مقدار ثابت سوم :
          دستورات مورد نظر;
          break;
 ......
 default:
     دستورات پیش فرض
 }
```


nastooh@aut.ac.ir

ساختارهای تصمیم

ساختار switch

- مثال: برنامه ای که یک عدد صحیح دریافت کرده و روز معادل با آن در هفته را نمایش دهد.

ابتدا دو راه با if:

```
if (day = = 1)
   printf (“Saturday”);
if (day = = 2)
   printf (“Sunday”);
if (day = = 3)
   printf (“Monday”);
if (day = = 4)
   printf (“Tuesdays”);
if (day = = 5)
   printf (“Wednesday”);
if (day = = 6)
   printf (“Thursday”);
if (day = = 7)
   printf (“Friday”);
if ( (day<1) || (day>7)
   printf (“Wrong Number”);
```

```
if (day = = 1)
   printf (“Saturday”);
else if (day = = 2)
        printf (“Sunday”);
     else if (day = = 3)
             printf (“Monday”);
          else if (day = = 4)
                  printf (“Tuesdays”);
               else if (day = = 5)
                       printf (“Wednesday”);
                    else if (day = = 6)
                            printf (“Thursday”);
                         else if (day = = 7)
                                 printf (“Friday”);
                              else
                                 printf(“Wrong Number”);
```

این دو راه را مقایسه کنید!

nastooh@aut.ac.ir

ساختارهای تصمیم

- ساختار switch

- مثال: برنامه ای که یک عدد صحیح دریافت کرده و روز معادل با آن در هفته را نمایش دهد.

با switch:

```
switch (number)
{
 case  1: printf(“Sat”);
          break;
 case  2: printf(“Sun”);
          break;
 case  3: printf(“Mon”);
          break;
 case  4: printf(“Tus”);
          break;
 case  5: printf(“Wed”);
          break;
 case  6: printf(“Thr”);
          break;
 case  7: printf(“Fri”);
          break;
default: printf(“Wrong No.”);
}
```


nastooh@aut.ac.ir

ساختارهای تصمیم

- ساختار switch

- در واقع در ساختار switch، همواره دستورات مربوط به default اجرا می شوند، مگر اینکه یکی از موارد مشخص شده در caseها رخ دهد!

- برای نوشتن دستورات مربوط به یک case، نیازی به استفاده از آکولاد نیست.

- استفاده از break در انتهای دستورات هر case **اجباری** است.
  - در صورت استفاده نکردن از break، در واقع case های بعدی با case مربوطه or می شوند. به عبارت دیگر، در ساختار switch، هنگام ورود از یک case، دستورات به ترتیب اجرا می شوند تا به یک break برسیم. (بی توجه به مقادیر case های بعدی)

- ساختار switch می تواند default نداشته باشد!

nastooh@aut.ac.ir

ساختارهای تصمیم

- ساختار switch
  - مثال: قطعه برنامه زیر را تحلیل کنید.

```
scanf("%d", &num);
switch (num % 4)
{
    case  0: printf("Yes");
             break;
    case  1:
    case  2:
    case  3: printf("No");
}
```

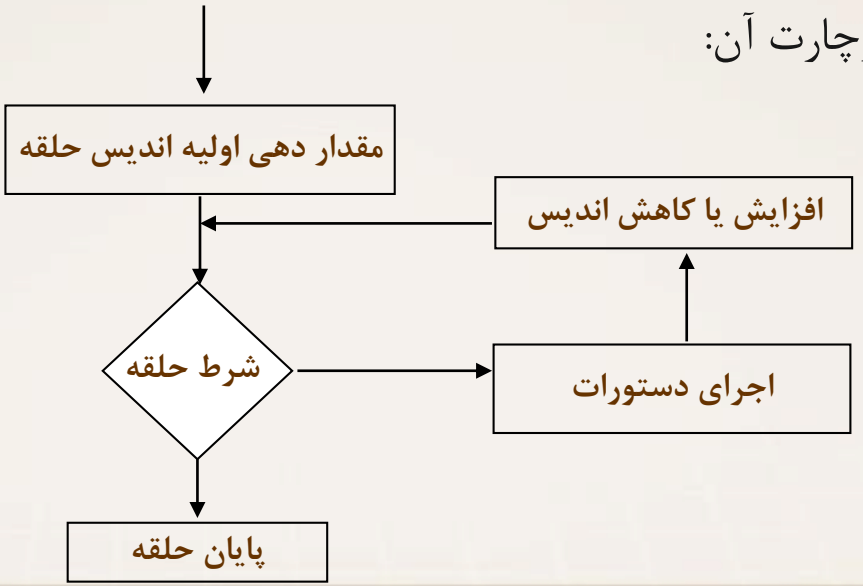

nastooh@aut.ac.ir

ساختارهای تکرار

- ساختار for:
  - هنگامی که عملیات تکراری باید به تعداد مشخصی تکرار شود، بهترین ساختار for است.
  - نحوهٔ استفاده:

```
for (گام حرکت ; شرط حلقه ; مقدار دهی اولیه اندیس حلقه)
{
   دستورات
}
```

  - فلوچارت آن:



nastooh@aut.ac.ir

ساختارهای تکرار

- ساختار for:
  - مثال: چاپ ۴ تا ستاره:

```
int  i;
for (i = 1;  i <= 4 ; i++)
{
    printf (“*”);
}
```

نتیجه:

****

  - مثال: چاپ اعداد یک تا ۱۰:

```
int  i;
for (i = 1; i <= 10  ; i++)
{
    printf (“%d ”, i);
}
```

نتیجه:

1 2 3 4 5 6 7 8 9 10

nastooh@aut.ac.ir

ساختارهای تکرار

- ساختار for:
  - مثال: چاپ اعداد زوج یک و دو رقمی:

```
int  i;
for (i = 2;  i <99  ; i = i + 2)
{
    printf (“%d ”, i);
}
```

نتیجه:

2 4 6 8 10 12 14 16 18 .... 94 96 98

- مثال: قطعه کد زیر را تحلیل کنید:

```
float  f;
for (f = 1;  f <= 3 ; f += 0.2)
{
    printf (“%f \n”, f);
}
```

*البته **توصیه نمیشود** برای کنترل حلقه ها از مقادیر اعشاری استفاده کنید! چون این مقادیر تقریبی هستند!*

nastooh@aut.ac.ir

ساختارهای تکرار

- ساختار for:
  - قطعه کد زیر را تحلیل کنید:

```
int  i;
for (i = 10;  i > 4  ; i--)
{
    printf (“%d ”, i);
}
```

نتیجه:

10 9 8 7 6 5

- مثال: قطعه کد زیر را تحلیل کنید:

```
int  i;
for (i = 10;  i >= 4  ; i--)
{
    printf (“%d ”, i);
}
```

نتیجه:

10 9 8 7 6 5 4

nastooh@aut.ac.ir

ساختارهای تکرار

- تمرین در کلاس:

- برنامه ای بنویسید که دو عدد را دریافت کرده و با کمک ضرب های متوالی، حاصل توان دو عدد را نمایش دهد.

- برنامه ای بنویسید که یک عدد را از ورودی دریافت کرده و فاکتوریل آن را نمایش دهد.

- برنامه ای بنویسید که یک عدد صحیح مثبت را دریافت کرده و کلیهٔ مقسوم علیه های آن را نمایش دهد.
  - یعنی برای ورودیِ ۱۲ باید اعداد: ۱ و ۲ و ۳ و ۴ و ۶ و ۱۲ را نمایش دهد.

- برنامه ای بنویسید که ۱۰ عدد را از ورودی دریافت کرده و بزرگترین آن ها را نمایش دهد.

nastooh@aut.ac.ir

ساختارهای تکرار

- حلقه های تو در تو (Nested Loop):
  - می توان حلقه ها را به صورت تو در تو استفاده کرد.
  - در این حالت، حلقه داخلی، خود به عنوان یک دستور از حلقه بیرونی محسوب می شود.
  - مثال: قطعه کد زیر را تحلیل کنید:

```
for (i = 1;  i <= 4  ; i++)
{
    printf (“%d”, i);
    for (j = 1; j <= 3; j++)
        printf(“*”);
    printf(“%d”, i);
}
```

خروجی:

1*12***23***34***4**

nastooh@aut.ac.ir

ساختارهای تکرار

- حلقه for تو در تو:
  - مثال: قطعه کد زیر را تحلیل کنید:

```
for (i = 1;  i <= 4  ; i++)
    for (j = 1; j <= 3; j++)
        printf("%d%d \n", i, j);
```

خروجی:

```
11
12
13
21
22
23
...
```


nastooh@aut.ac.ir

ساختارهای تکرار

- حلقه for تو در تو:
  - مثال: قطعه کد زیر را تحلیل کنید:

```
for (i = 1;  i <= 5  ; i++)
{
   for (j = 1; j <= 8; j++)
        printf("*");
   printf("\n");
}
```

خروجی:

```
********
********
********
********
********
```


nastooh@aut.ac.ir

ساختارهای تکرار

- حلقه for تو در تو:
  - مثال: قطعه کد زیر را تحلیل کنید:

```
for (i = 1;  i <= 5  ; i++)
{
   for (j = 1; j <= i; j++)
        printf(“*”);
   printf(“\n”);
}
```

خروجی:

```
*
**
***
****
*****
```


nastooh@aut.ac.ir

ساختارهای تکرار

- حلقه for تو در تو:
  - مثال: قطعه کد زیر را تحلیل کنید:

```
for (i = 5;  i <= 1  ; i--)
{
   for (j = 1; j <= i; j++)
        printf(“*”);
   printf(“\n”);
}
```

خروجی:

```
*****
****
***
**
*
```


nastooh@aut.ac.ir

ساختارهای تکرار

- حلقه for تو در تو:
  - مثال: قطعه کد زیر را تحلیل کنید:

```
for (i = 1;  i <= 5  ; i++)
{
   for (j = 2; j <= i; j++)
        printf(“   ”);
   for (k = 5; k >= i; k--)
        printf(“*”);
   printf(“\n”);
}
```

⟺

```
for (k = 1; k <= 6 - i;  k++)
```

خروجی:

```
*****
 ****
  ***
   **
    *
```


nastooh@aut.ac.ir

ساختارهای تکرار

- حلقه for تو در تو:
  - تمرین در منزل: سعی کنید اشکال زیر را جداگانه در خروجی نمایش دهید:

```
*********
*******
*****
***
*
```

```
*
***
*****
*******
*****
***
*
```

```
*
**
***
****
*****
```

```
*
***
*****
*******
*********
```


nastooh@aut.ac.ir

ساختارهای تکرار

- حلقه for تو در تو:

- تمرین: برنامه ای بنویسید که جدول ضرب ۱۰ در ۱۰ را با فرمت مناسبی نمایش دهد.
  - راهنمایی ۱، مثلا به صورت شکل زیر!
  - راهنمایی ۲، برای چاپ منظم خروجی از تعیین طول میدان مقادیر خروجی در printf استفاده کنید!
  - راهنمایی ۳، از چاپ \n در جای مناسب غافل نشوید!

```
  1   2   3   4   5   6   7   8   9  10
  2   4   6   8  10  12  14  16  18  20
  3   6   9  12  15  18  21  24  27  30
  4   8  12  16  20  24  28  32  36  40
  5  10  15  20  25  30  35  40  45  50
  6  12  18  24  30  36  42  48  54  60
  7  14  21  28  35  42  49  56  63  70
  8  16  24  32  40  48  56  64  72  80
  9  18  27  36  45  54  63  72  81  90
 10  20  30  40  50  60  70  80  90 100
```


nastooh@aut.ac.ir

ساختارهای تکرار

- حلقه for:
  - هر کدام از قسمت های سه گانهٔ حلقهٔ for را می توان خالی نگهداشت!! مثال:

```
for ( ; ; )
```

  - اگر قسمت شرط حلقهٔ for خالی باشد، عملاً حلقهٔ بی نهایت تشکیل می شود!!!
  - اگر بعد از حلقهٔ for از سمی کالن استفاده کنیم، بدنهٔ حلقه تهی می شود. یعنی حلقه به تعداد نیاز میچرخد، اما عملا هیچ کاری انجام نمی شود. مثال:

```
for (i = 1; i <= x ; i++) ;
printf (“%d”, i);
```

  - می توان در هر یک از بخش های سه گانهٔ حلقهٔ for با استفاده از کاما چندین دستور را قرار داد. اگر در قسمت شرط چند شرط قرار گیرد، فقط و فقط شرط سمت راست ملاک خواهد بود. مثال:

```
for (i=1, j=4, k=10   ;   i<5, j<8   ;   i++, j--, k=k-2)
```

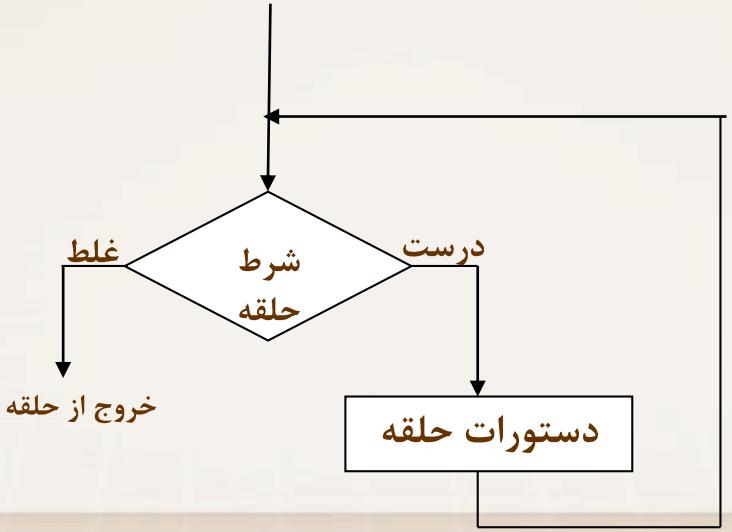

nastooh@aut.ac.ir

ساختارهای تکرار

- ساختار while:
  - معمولا وقتی تعداد دفعات تکرار یک حلقه از پیش مشخص نیست و فقط به یک شرط وابسته است، بهترین ساختار while است.
  - نحوهٔ استفاده:

```
while (شرط حلقه)
{
    دستورات
}
```

  - فلوچارت:



nastooh@aut.ac.ir

ساختارهای تکرار

- ساختار while:
  - مثال ساده: نمایش اعداد زوج تک رقمی با while

```
int  i = 2;

while (i < 9)
{
     printf(“%d  ”, i);
     i += 2;
}
```


nastooh@aut.ac.ir

ساختارهای تکرار

- تمرین در کلاس:

  - برنامه ای بنویسید که یک عدد صحیح مثبت را از ورودی دریافت کرده و مشخص کند آن عدد چند رقمی است.

nastooh@aut.ac.ir

ساختارهای تکرار

- ساختار do-while:
  - همانند حلقهٔ while است، اما شرط حلقه در انتهای آن چک می شود. پس بدنهٔ حلقه حتماً یکبار اجرا خواهد شد.
  - نحوهٔ استفاده:

```
do
{
    دستورات
} while (شرط حلقه);
```

- آیا می توانید فلوچارت ساختار do-while را بکشید؟

nastooh@aut.ac.ir

ساختارهای تکرار

- ساختار do-while:
  - مثال ساده: نمایش اعداد زوج تک رقمی با do-while

```
int i = 2;
do
{
   printf(“%d  ”, i);
   i += 2;
} while (i < 9);
```


nastooh@aut.ac.ir

ساختارهای تکرار

- نکته:
  - برای ایجاد حلقهٔ بی نهایت می توان از شرط زیر استفاده کرد:

```
while ( 1 )
{
}
```

می دانیم که در زبان C در واقع شرط حلقهٔ فوق، به صورت زیر تفسیر می شود:

```
while ( 1 != 0 )
{
}
```

که همواره صحیح است!

nastooh@aut.ac.ir

کنترل غیر شرطی

- دستور break:
  - این دستور برای خاتمه دادن به حلقه استاده می شود!
  - نحوهٔ استفاده:

```
break;
```

  - بعد از break، دستورات بعد از حلقه اجرا می شوند.
  - مثال: قطعه کد زیر را تحلیل کنید:

```
for (i = 1 ; i <= 10 ; i++)
{
   printf (“%d”, i);
   break;
   printf (“%d”, i);
}
```

خروجی:

1

nastooh@aut.ac.ir

کنترل غیر شرطی

- دستور break:
  - با استفاده از if، می توان استفادهٔ موثرتری از break برای خاتمه دادنِ هوشمندانه به حلقه ها کرد.
  - مثال: برنامه ای بنویسید که تا زمانی که یک عدد منفی از وارد شود، تعدادی عدد را از ورودی خوانده و مجموع آنها را نمایش دهد.

```
int  num, sum = 0;
while (1)
{
   scanf("%d", &num);
   if (num < 0)
       break;
   sum += num;
}
printf ("sum= %d ", sum);
```


nastooh@aut.ac.ir

کنترل غیر شرطی

- دستور break:
  - در حلقه های تو در تو، دستور break فقط باعث شکستن داخلی ترین حلقه خواهد شد.
  - مثال: تحلیل کنید.

```
for (i = 1; i <= 5 ; i++)
{
   for( j = 1; j <= 5 ; j++)
   {
        printf (“%d”, j);
        if ( j = = 3)
            break;
   }
   printf (“\n”);
}
```

خروجی:

```
123
123
123
123
123
```


nastooh@aut.ac.ir

کنترل غیر شرطی

- دستور break:

- مثال: تحلیل کنید.

```
for (i = 1; i <= 5 ; i++)
{
   for( j = 1; j <= 5 ; j++)
   {
        printf (“%d”, j);
        if ( i = = 3)
            break;
   }
   printf (“\n”);
}
```

خروجی:

```
12345
12345
1
12345
12345
```


nastooh@aut.ac.ir

کنترل غیر شرطی

- دستور continue:
  - این دستور برای خاتمه دادن به یک دور حلقه، و شروع دور بعد به کار می رود.
  - نحوهٔ استفاده:

continue;

  - در واقع بعد از اجرای continue، از مابقی دستورات حلقه صرف نظر شده و دور بعدی شروع می شود.
  - مثال:

```
for (i = 1 ; i <= 10 ; i++)
{
   printf (“%d”, i);
   continue;
   printf (“%d”, i);
}
```

خروجی:

12345678910

nastooh@aut.ac.ir

کنترل غیر شرطی

- دستور continue:
  - در حلقه های تو در تو، دستور continue فقط بر روی یک حلقه و آن هم داخلی تین حلقه می کند.
  - مثال: تحلیل کنید.

```
for (i = 1; i <= 5 ; i++)
{
   for( j = 1; j <= 5 ; j++)
   {
      if ( j = = 3)
         continue;
      printf (“%d”, j);
   }
   printf (“\n”);
}
```

خروجی:

```
1245
1245
1245
1245
1245
```


nastooh@aut.ac.ir

کنترل غیر شرطی

- دستور continue:
  - در حلقه های تو در تو، دستور continue فقط بر روی یک حلقه و آن هم داخلی تین حلقه می کند.
  - مثال: تحلیل کنید.

```
for (i = 1; i <= 5 ; i++)
{
   for( j = 1; j <= 5 ; j++)
   {
      if ( i = = 3)
         continue;
      printf (“%d”, j);
   }
   printf (“\n”);
}
```

خروجی:

```
12345
12345

12345
12345
```


nastooh@aut.ac.ir

کنترل غیر شرطی

- دستور goto:
  - کنترل برنامه را به نقطه ای مشخص منتقل می کند.
  - نحوهٔ استفاده:

goto label;

این دستور، کنترل را به یکی از دستورات که با برچسبِ label مشخص شده است، منتقل می کند.

- یکی از دستورات باید حتما برچسب label داشته باشد. مثال صفحه بعد را ببینید!

- **امروزه از goto بسیار به ندرت استفاده می شود.**

nastooh@aut.ac.ir

کنترل غیر شرطی

- دستور goto:
  - می توان تمام حلقه ها را با ترکیب if و goto پیاده سازی کرد.

- مثال:

```
...
...
      i  = 1;
xx:  printf (“%d”, i);
      i++;
      if (i < 10)
         goto  xx;
...
...
```

- در برنامه های پیچیده تر، این کار به شدت خوانایی را **کاهش داده** و اشکال زدایی برنامه را **مشکل می کند**. (به چنین کدهایی، کد اسپاگتی گویند!!! که باید از آنها به کلی **اجتناب کرد**.)

nastooh@aut.ac.ir

کنترل غیر شرطی

- دستور goto:

- تقریباً تنها جایی که می توان از goto استفاده کرد، خروجی از چندین حلقهٔ تو در تو است. زیرا break فقط باعث شکستن حلقهٔ داخلی می شود.

```
for (...)
{
   for (...)
   {
       for(...)
       {
           for(...)
           {
               if (شرط)  goto  label5;
            }
        }
    }
}
label5: ....
```


nastooh@aut.ac.ir

کنترل غیر شرطی

- تابع exit():
  - اجرای این تابع باعث بسته شدن کل برنامه می شود.
  - الگوی آن در فایل stdio.h قرار دارد.

- نحوهٔ استفاده:

```
exit( );
```

- می توان در جای مناسب و با وقوع شرایط خاص، از این تابع استفاده کرد!
  - زمانی از این کار استفاده می کنیم که با وقوع یک شرط خاص، دیگر ادامهٔ برنامه بی معنی است و برنامه باید خاتمه یابد.

nastooh@aut.ac.ir

یادآوری چند عادت مهم در برنامه نویسی

- عادت اول:
  - رعایت تورفتگی ها (دندانه گذاری)
- عادت دوم:
  - ثبت کامنت های مناسب و خوانا
- عادت سوم:
  - استفاده از اسامی با معنی

این سه عادت را ملکهٔ ذهن خود کنید.

nastooh@aut.ac.ir

پایان

nastooh@aut.ac.ir