برنامه نویسی پیشرفته
به زبان
C(++)

مدرس

امير جليلي ايراني
www.amirjalili.ir
amirjaliliirani@gmail.com

تحصیلات

لیسانس کامپیوتر – نرم افزار
دکتری و فوق لیسانس کامپیوتر – هوش مصنوعي
از دانشگاه علوم و تحقیقات تهران

هوش مصنوعي
Artificial Intelligence
(A. I.)

- هوش مصنوعي:

الهام از موجودات زنده به ويژه انسان براي ايجاد قابليت هاي استدلال، استنتاج، يادگيري و رفتار هوشمند.

- گرايش هوش مصنوعي:

عبارت است از هوشمند سازي كامپيوترها و سيستم هاي مبتني بر كامپيوتر.

مرجع درسي

برنامه نويسي به زبان c و c++

علوم رايانه

تاليف

عين الله جعفر نژاد قمي

مرجع كنكوري

برنامه نويسي C و C++

انتشارات گسترش علوم پايه

تاليف

حميد رضا مقسمي

سر فصلها

- جايگاه برنامه سازي، روشهاي طرح برنامه و چرخه حيات.
- معرفي كامل زبان C.
- ايده شيءگرايي.
- برنامه سازي شيءگرا.
- اشياء، چندريختي، وراثت و كپسوله سازي.
- معرفي زبان C++.
- آزمون و مستند سازي.

پيش نيازها

- مباني كامپيوتر و برنامه نويسي.
- الگوريتم و فلوچارت.

دليل اهميت درس

- يادگيري زبان برنامه نويسي سيستمي C.
- كاربردي و باعث تقويت برنامه نويسي.
- پايه و پيش نياز بسياري از دروس تخصصي .
- مطرح در كنكور كارشناسي ارشد گرايش هاي مختلف.

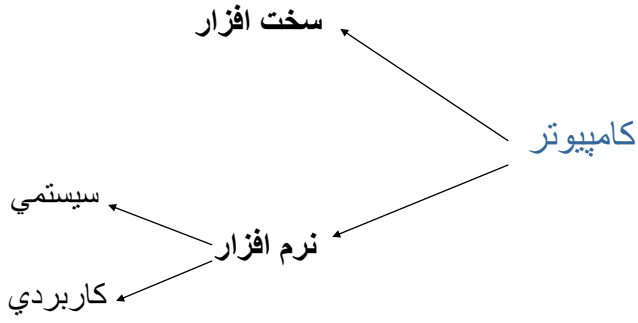
سیستم ارزشیابی

☐ کوئیز	۱
☐ تمرینات	۱
☐ میان ترم	۴
☐ پایان ترم	۱۴
☐ حضور مرتب و منظم	مازاد
☐ پروژه	اختیاري

برنامه‌سازی پیشرفته

جلسهٔ اول

برنامه‌سازی پیشرفته



s.w. = program + data + document

amirjalili.ir

برنامه‌سازی پیشرفته

- برنامه:

مجموعه ای از دستورات که برای انجام مساله معینی به کامپیوتر داده میشود.

- برنامه‌سازی یا برنامه نویسی:

عبارت است از نوشتن برنامه.

- زبان برنامه نویسی:

مجموعه علائم و قواعد و سمبولها برای نوشتن برنامه کامپیوتری.

برنامه‌سازی پیشرفته

وظیفه یک برنامهٔ کامپیوتری

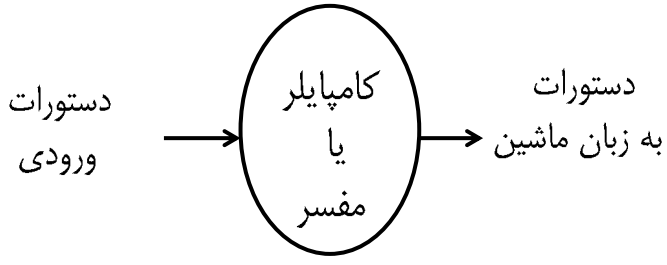
برنامه‌سازی پیشرفته

سطوح زبانها

- زبانهاي سطح بالا يا HLL
- زبانهاي سطح مياني يا MLL
- زبانهاي سطح پايين يا LLL

برنامه‌سازی پیشرفته

کامپایلر



amirjalili.ir

برنامه‌سازی پیشرفته

سبک برنامه نویسي

- بالا به پایین یا top – down
  - روش تقسیم و غلبه.
  - معمولاً داراي الگوريتم هاي بازگشتي.
- پایین به بالا یا button up
  - روش پویا.
  - معمولاً داراي الگوريتم هاي غير بازگشتي.

برنامه‌سازی پیشرفته

ساختمان داده ها

- ایستا

۱- اولیه: انواع داده اولیه ۲- غیر اولیه: آرایه - استراکچر

- نیمه ایستا

صف – پشته

- پویا

۱- خطي: ليست هاي پيوندي ۲- غير خطي: درخت - گراف

amirjalili.ir

برنامه‌سازی پیشرفته

الگوريتم

روشي گام به گام براي حل مساله كه:

1. داراي شروع و پايان مشخص.
2. ورودي و خروجي مشخص.
3. پايان پذير يا متناهي.(بر خلاف برنامه)
4. واضح و بدون ابهام.
5. قابل بيان به زبان همه فهم و طبيعي.
6. ترتيب و توالي اجراي دستورات مشخص باشد.

برنامه‌سازی پیشرفته

طراحي الگوريتم

فراگيري روش ها و تكنيک هاي الگوريتم نويسي
براي حل مسايل مختلف و كلاسيک

- روش تقسیم و غلبه — divide and conquer
- روش پویا — dynamic
- روش حریصانه — greedy
- روش عقبگرد — backtracking
- روش شاخه و حد — branch and bound

برنامه‌سازی پیشرفته

شبه کد یا pseudo code

- اشكالات بيان الگوريتم به زبان عادي:

1. دشوار بودن بيان الگوريتم هاي پيچيده با اين روش.
2. دشوار بودن تبديل اين بيان به يک برنامه كامپيوتري.

- چرا شبه كد:

1. فارغ بودن از پيچيدگي هاي زبان هاي برنامه نويسي.
2. راحتي در قابليت تبديل به برنامه كامپيوتري

برنامه‌سازی پیشرفته

چرخه حیات

- از زماني كه سيستمي **تعريف، ساخته و نگهداري** ميشود تا زمانيكه بنا به هر دليلي نگهداري نشود.

Difination-Development-Support

- از لحظه اي كه نياز بوجود مي آيد تا زمانيكه نياز رفع گردد.

برنامه‌سازی پیشرفته

amirjalili.ir

آزمون یا تست

- حضور خطا را مشخص می سازد .

عدم حضور خطا را مشخص نمی سازد.

مستند سازی

- هنگام پایان هر طرحی اینکار انجام میگیرد.

هیچ طرحی بدون مستند سازی کامل نیست.

برنامه‌سازی پیشرفته

جلسهٔ دوم

برنامه‌سازی پیشرفته

سابقهٔ تاریخی زبان C

- زبان B
- زبان BCPL
- زبان C :

در سال ۱۹۷۲ توسط دنیس ریچی طراحی شد.

برنامه‌سازی پیشرفته

ویژگیهای بارز زبان C

1. C یک زبان میانی است.
2. C یک زبان ساختیافته است.
3. C زبان برنامه‌نویسی سیستم یا همه منظوره است.
4. C یک زبان قابل حمل است.
5. C زبانی قابل انعطاف و قدرتمند است.

برنامه‌سازی پیشرفته

کلیات زبان C

- حساس به حروف (Case Sensitive)
  INT و int
- کلمات کلیدی کم(reserved word)
  مثال: for ، if ، while
  نکته: اکثر کلمات کلیدی با حروف کوچک هستند.

برنامه‌سازی پیشرفته

کلیات زبان C

- ; جدا کنندهٔ دستورات از یکدیگر:
  هر دستور در یک یا چند سطر
  چند دستور در هر سطر

amirjalili.ir

برنامه‌سازی پیشرفته

کلیات زبان C

- توضیحات بین */ و /*یا بعد از //

```
/* this is a sample comment
example. */
// this is sample comment.
```

برنامه‌سازی پیشرفته

استانداردسازی زبان C

- گونه‌های مختلف زبان C
- استانداردسازی زبان C :

ANSI C

برنامه‌سازی پیشرفته

کامپایلر پیشنهادی زبان C

Borland C++ 3.1
Turbo c++

برنامه‌سازی پیشرفته

مجموعهٔ دستورات هر زبان برنامه‌نویسی

- دستورات کامپایلر زبان
- دستورات ورودی – خروجی
- دستورات محاسباتی و منطقی
- دستورات کنترل روند اجرای برنامه

amirjalili.ir

برنامه‌سازی پیشرفته

Data type

انواع داده‌های اصلی

int
float
double
char
void
boolean , string ?!!

برنامه‌سازی پیشرفته

int

اعداد صحیح با دامنه محدود

برای کامپیوترهای شخصی دو بایت

+۳۲۷۶۷

-۳۲۷۶۸

برنامه‌سازی پیشرفته

float

اعداد حقیقی با دامنهٔ محدود

نمایش معمولی

نمایش علمی

12.3E- 4 = 12.00003

برنامه‌سازی پیشرفته

double

اعداد حقیقی با دقتی بیشتر از **float**

برنامه‌سازی پیشرفته

char

کاراکترها نمادها یا حروف

‘a’
‘A’
‘+’
‘~’

بسته به محل استفاده عدد یا کاراکتر است.

برنامه‌سازی پیشرفته

void

دادهٔ تهی

دارای کاربردهای مختلف

مثال: توابع فاقد خروجی

amirjalili.ir

برنامه‌سازی پیشرفته

انواع دادهای دیگر

با ترکیب کلمات زیر با برخی از انواع داده‌های اصلی:

signed ، **unsigned** (با علامت ، بدون علامت)

long ، **short**

مانند:

unsigned int

long int

unsinged long int

برنامه‌سازی پیشرفته

نوع داده	مقادیر	حافظه لازم
int	32767 تا -32768	۲ بایت
unsigned int	0 تا 65535	۲بایت
long int	2147483647 تا -2147483648	۴ بایت
unsigned long int	0 تا 4294967295	۴ بایت
char	یك كاركتر	۱بایت
unsigned char	127 تا -128	۱ بایت
float	1.2e-38 تا 3.4e38	۴ بایت
double	2.2e-308 تا 1.8e308	۸ بایت

amirjalili.ir

برنامه‌سازی پیشرفته

متغیرها یا توابع

قوانین نامگذاری متغیرها:

- حروف ’a‘ تا ’z‘ ، ’A‘ تا ’Z‘ ، ارقام و ’_‘
- اولین کاراکتر حرف باشد.
- کلمات کلیدی نمی‌توانند نام متغیر باشند.

برنامه‌سازی پیشرفته

متغیرها

اسامی مجاز:

count

c124

avg_grade

اسامی غیرمجاز:

1test

bin#tree

for

amirjalili.ir

برنامه‌سازی پیشرفته

تعریف متغیر

نوع داده نام متغیر ;

```
int x ;
float m, n ;
char ch1, ch2, ch3 ;
long int count ;
```

برنامه‌سازی پیشرفته

مقدار دهی اولیه به متغیرها

```
int  x = 5, y ;

char ch1 = ‘a’, ch2 = ‘A’,ch ;
```


برنامه‌سازی پیشرفته

ثابتها

تعریف ثابت:

```
#define  نام ثابت      مقدار ثابت
```

یا

```
const    نوع داده    نام ثابت = مقدار;
```

برنامه‌سازی پیشرفته

مثال

```
#define   M    100
#define   P    3.14

const   int    n = 100 ;
const   char   c = ‘a’ ;
```


برنامه‌سازی پیشرفته

عملگرها

- محاسباتی
- رابطه‌ای
- منطقی
- بیتی

برنامه‌سازی پیشرفته

عملگرهای محاسباتی

- (یکانی) -

+ , - , * , / , %

++ , --

برنامه‌سازی پیشرفته

مثال

```
- x
x + y
x / y
x % y
```

برنامه‌سازی پیشرفته

++ و --

تفاوت

x ++

و

++ x

برنامه‌سازی پیشرفته

عبارات محاسباتی

ترکیبی از متغیرها، ثابتها و عملگرهای محاسباتی

```
x + y * z / 2 - y
```

برنامه‌سازی پیشرفته

دستور انتساب

عبارت محاسباتی یا مقدار ثابت = نام یک متغیر;

```
int x, y = 19, z ;
x = 10 ;
z = x * 2 + y ;
```


amirjalili.ir

برنامه‌سازی پیشرفته

اولویت عملگرها

```
w = x * y + w
```

?

```
w = (x * y) + w
```

یا

```
w = x * (y + w)
```

برنامه‌سازی پیشرفته

قواعد اولویت عملگرها و پرانتزها

```
w = x * y + w
```

تقدم عملگرهای محاسباتی:

```
()
++  --
- (یکانی)
*  /  %
+  -
```


برنامه‌سازی پیشرفته

تبدیل انواع

```
char ch ;
int i ;
float f, result; ...
result = (ch / i) + f ;
ch = i ;
i = result ;
```

برنامه‌سازی پیشرفته

عملگرهای رابطه‌ای

> , >= , < , <=

== , !=

مثال:

x > y

x == y

x != y

برنامه‌سازی پیشرفته

amirjalili.ir

عملگرهای رابطه‌ای

حاصل عملگرهای رابطه ای درست **T** یا غلط **F** میباشد

از آنجائیکه در **C** عملگر منطقی نداریم

هر متغیر از هرنوع با هر مقداری:

0 ➔ **F**

غیر صفر ➔ **T**

برنامه‌سازی پیشرفته

عملگرهای منطقی

!

&&

||

مثال:

(x > 10) && (x < y)

! (x > 20)

برنامه‌سازی پیشرفته

amirjalili.ir

عملگرهای بیتی

(And) &

(Or) |

(Xor) ^

(Not) ~

(Right Shift) >>

(Left Shift) <<

برنامه‌سازی پیشرفته

مثال

```
char x = 7 , y ;
y = x & 2 ;
y = ~y ;
```


amirjalili.ir

برنامه‌سازی پیشرفته

عملگرهای دیگر

- عملگرهای ترکیبی شامل:

  `+= , -= , *= , /= , %=`

  که `x += y` معادل `x = x + y`

- غیره (در جای خود توضیح داده خواهند شد)

برنامه‌سازی پیشرفته

جلسهٔ سوم

برنامه‌سازی پیشرفته

انواع خطاهای برنامه نویسی:

۱- خطای گرامری یا syntax error

۲- خطای منطقی یا logical error

۳- خطای حین اجرا یا run time error

برنامه‌سازی پیشرفته

تفاوت خطا و هشدار :

- خطا یا error
- هشدار یا warning

amirjalili.ir

برنامه‌سازی پیشرفته

انواع توابع در زبان C:

۱- تابع اصلی یا main

۲- توابع از پیش تعریف شده یا کتابخانه ای مانند printf، pow ...

۳- توابع فرعی که کاربر در صورت نیاز تعریف نماید.

برنامه‌سازی پیشرفته

ساختار یک برنامهٔ ساده

```
#include  < header file>
void    main()
{
clrscr();              ➔ <conio.h>
تعریف متغیرها
دستورات اجرایی
getch();               ➔ <conio.h>
}
```


amirjalili.ir

برنامه‌سازی پیشرفته

توابع ورودی - خروجی C

تابع و نه دستور

مهمترین: printf و scanf

<stdio.h>

برنامه‌سازی پیشرفته

تابع خروجی printf

printf (“ عبارت ۱ ” , عبارت ۲);

عبارت ۲ : اطلاعات یا متغییرهایی که باید به خروجی منتقل شوند.

(اختیاری است)

amirjalili.ir

برنامه‌سازی پیشرفته

printf (“ عبارت ۱ ” , عبارت ۲);

عبارت ۱ می‌تواند شامل:

اطلاعات یا پیغام هایی که باید عینا در خروجی چاپ شوند

کاراکترهای تعیین کنندهٔ فرمت خروجی

کاراکترهای کنترلی

برنامه‌سازی پیشرفته

کاراکترهای تعیین کنندهٔ فرمت خروجی

- مشخص کنندهٔ نوع اطلاعات ذکر شده در عبارت ۲
- با علامت % شروع می‌شوند. مانند:

%c (برای کاراکتر)

%d (برای عدد صحیح)

%f (برای عدد اعشاری)

برنامه‌سازی پیشرفته

کاراکترهای کنترلی

- تعیین شکل اطلاعات خروجی
- با علامت \ شروع می‌شوند. مانند:

\n انتقال به سطر جدید

\f انتقال به صفحهٔ جدید

برنامه‌سازی پیشرفته

مثال

```
printf (“this is a test.”);
```

خروجی

```
this is a test.
```


برنامه‌سازی پیشرفته

مثال

```
int  i =10 ;
char ch = ‘a’ ;
printf (“%d , %c” , i , ch);
```

خروجی

```
10 , a
```

برنامه‌سازی پیشرفته

مثال

```
int i =10 ;
char ch = ‘a’ ;
printf (“i = %d , ch = %c” , i , ch);
```

خروجی

```
i = 10 , ch = a
```


amirjalili.ir

برنامه‌سازی پیشرفته

مثال

```
int i =10 ;
char ch = ‘a’ ;
printf (“i = %d\nch = %c” , i , ch);
```

خروجی

```
i = 10
ch = a
```

برنامه‌سازی پیشرفته

اولین برنامه

```
#include  <conio.h>
#include  <stdio.h>
void  main()
{
printf (“This is our first C program”) ;
getch();
}
```


برنامه‌سازی پیشرفته

amirjalili.ir

تابع ورودی scanf

scanf (“ عبارت ۱ ” , عبارت ۲);

عبارت ۲ : آدرس متغیرهایی که باید خوانده شوند

عبارت ۱ : نوع متغیرها و نحوهٔ خوانده شدن آنها

برنامه‌سازی پیشرفته

عبارت ۱ شامل:

1. کاراکترهای فرمت. مشخص‌کنندهٔ نوع اطلاعات.

مانند:

%c (کاراکتر)

%d (عدد صحیح)

برنامه‌سازی پیشرفته

amirjalili.ir

2. کاراکتر فضای خالی

تاثیر: در نظر نگرفتن (رد کردن) فضای خالی در اطلاعات ورودی

3. کاراکترهای دیگر

تاثیر: خواندن و صرفنظر کردن از کاراکتر فوق

برنامه‌سازی پیشرفته

مثال

```
int i , j ;
char ch ;
scanf (“%d%d%c” , &i , &j , &ch) ;
```


amirjalili.ir

برنامه‌سازی پیشرفته

مثال برنامه‌نویسی

دریافت شعاع یک دایره از ورودی و چاپ
مساحت آن در خروجی

برنامه‌سازی پیشرفته

```
#include  <stdio.h>
#include  <conio.h>
void  main()
{
float r, area ;
```


برنامه‌سازی پیشرفته

```
  printf ("Enter the radius:") ;
  scanf ("%f" , &r) ;
  area = 3.14 * r * r ;
  printf ("\n area = %f" , area) ;
  getch();
}
```

برنامه‌سازی پیشرفته

- تابع cout :

```
Cout<<"string";
Cout<<variables;
```

- تابع cin :

```
Cin>>variables;
```

```
<iostream.h>
```


انواع توابع ورودي و خروجي

```
char ch;
ch=getch();
```

يا

```
scanf("%c",&ch);
```

يا

```
cin>>ch;
```

برنامه‌سازی پیشرفته

جلسهٔ چهارم

برنامه‌سازی پیشرفته

دستورات کنترل روند برنامه

ساختارهای تصمیم

حلقه‌های تکرار

برنامه‌سازی پیشرفته

ساختارهای تصمیم

if

if else

switch case

برنامه‌سازی پیشرفته

```
if (شرط یا عبارت منطقی)
    دستور ;
```

مثال:

```
if (x > 10)
    x ++ ;
```

برنامه‌سازی پیشرفته


```
if  (عبارت منطقی)
{
      دستور ۱ ;
      دستور ۲ ;
      ...
      دستور n ;
}
```


برنامه‌سازی پیشرفته

amirjalili.ir

مثال:

```
if  (x < y)
{
  x = x + y ;
  cout<<x << y ;
}
```

برنامه‌سازی پیشرفته

```
if  (عبارت منطقی)
{
   Aدستور ;
}
else
{
   Bدستور ;
}
```


برنامه‌سازی پیشرفته

```
if  ( (x > 10) && (x < 20) )
 {
      x = x + 1 ;
      y ++ ;
}
else
{
      x= 3 + x;
}
```

برنامه‌سازی پیشرفته

- مثالی از نحوه نوشتن شرط:

متغیر x یکی از مقادیر زیر را میتواند داشته باشد:

X = { 0 ➔ F ➔ A دستورات
غیر صفر ➔ T ➔ B دستورات

برنامه‌سازی پیشرفته

```
if (x==0)
{
    Aدستور ;
}
else
{
    Bدستور ;
}
```

```
if (!x)
{
    Aدستور ;
}
else
{
    Bدستور ;
}
```

```
if !((x<0)||(x>0))
{
    Aدستور ;
}
else
{
    Bدستور ;
}
```

برنامه‌سازی پیشرفته

```
if ((x<0)||(x>0))
{
    دستورB ;
}
else
{
    دستورA ;
}
```

```
if (x)
{
    دستورB ;
}
else
{
    دستورA ;
}
```

```
if (x!=0)
{
   دستورB ;
}
else
{
   دستورA ;
}
```


amirjalili.ir

برنامه‌سازی پیشرفته

مثال برنامه‌نویسی

برنامه‌ای بنویسید که عددی را از ورودی دریافت کرده و زوج یا فرد بودن آن را مشخص کند.

برنامه‌سازی پیشرفته

```
#include  <stdio.h>
#include  <conio.h>
void   main()
 {
        int i ;
        cin>>i ;
        if  (i % 2 == 0)
              cout<<"The number is even." ;
        else
              cout<<"The number is   odd.";
        getch();
}
```


برنامه‌سازی پیشرفته

دستورات شرطی متداخل(تو در تو)

```
if  ...
      ...
else   if  ...
            ...
         else   if ...
```

برنامه‌سازی پیشرفته

مثال:

```
if  (ch == '+')
      r = x + y ;
else  if (ch == '-')
          r = x - y ;
      else if (ch == '*')
              r = x * y ;
```


amirjalili.ir

برنامه‌سازی پیشرفته

دستور switch

برای تصمیم‌گیریهای چندگانه بر اساس
مقادیر مختلف یک عبارت
(به جای if های متداخل)

برنامه‌سازی پیشرفته

```
switch  (متغیر یا عبارت)  {
   case   مقدار ۱ :
                  دستورات ۱
                  break ;
   case   مقدار ۲ :
                  دستورات ۲
                  break ;
```


برنامه‌سازی پیشرفته

```
        .
        .
        .
   default :
                  دستورات n;
}
```

مثال

```
char  ch;
 int x,y,r;
cin>>x>>y;
ch=getch();//cin>>ch;
switch  (ch)  {
   case ‘+’ :
          r = x + y ;
          break ;
```


برنامه‌سازی پیشرفته

```
   case ‘-’ :
          r = x - y ;
          break ;
   case ‘*’ :
          r = x * y ;
          break ;
```

برنامه‌سازی پیشرفته

```
    case '/' :
            r = x / y ;
            break ;
    default :
            r = 0 ;
            cout<<"invalid operator" ;
}
cout>>r;
```


amirjalili.ir

برنامه‌سازی پیشرفته

نکات:

- بخش **default** اختیاری است.
- مقادیر موجود در **case** ها نباید مساوی باشند.

برنامه‌سازی پیشرفته

- در switch فقط تساوی را می‌توان چک کرد.
- در صورت عدم استفاده از break دستورات case های بعدی تا آخر اجرا خواهد شد.

برنامه‌سازی پیشرفته


```
int grade ;
switch  ( grade )  {
      case 18 :
      case 19 :
      case 20 :
                  cout<<"Good" ;
                  break ;
```

برنامه‌سازی پیشرفته

```
case 10 :
        cout<<“Acceptable”;
}
```


برنامه‌سازی پیشرفته

ساختارهای کنترل غیرشرطی

- **break** (پرش به اولین دستور بعد از بدنه یا حلقه)
- **continue** (برگشت به اولین دستور بدنه یا حلقه)
- **goto** (پرش به برچسب مدنظر)

بدلیل پایین آوردن خوانایی استفاده از آنها توصیه نمی‌شود.

برنامه‌سازی پیشرفته

جلسهٔ پنجم

برنامه‌سازی پیشرفته

- **انواع ساختارهای تکرار:**

۱- حلقه های معین:

حلقه هایی که تعداد تکرار آن از پیش معین باشد.

۲- حلقه های نا معین:

حلقه هایی که تعداد تکرار آن از پیش معین نباشد بلکه بسته به کاربر آن تعداد متفاوتی میتواند داشته باشد.

برنامه‌سازی پیشرفته

ساختارهای تکرار

for

while

do ... while

برنامه‌سازی پیشرفته

حلقه for

یکی از قویترین و کاملترین دستورات C

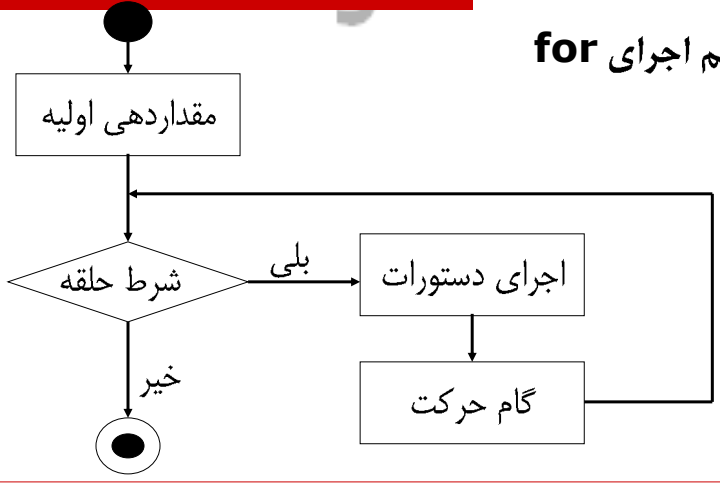
در تمامی کاربردهای حلقه می‌تواند بکار رود

برنامه‌سازی پیشرفته


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


برنامه‌سازی پیشرفته

الگوریتم اجرای for



مثال

```
int sum=0,i,num;
for ( i = 1 ; i < 10 ; i ++ )
{
cin>>num;
sum = sum + num ;
}
```


برنامه‌سازی پیشرفته

نکات: هر یک از ۳ قسمت فوق اختیاری هستند.

مثال: حلقه بینهایت

```
for ( ; ; )
{
...
}
```

برنامه‌سازی پیشرفته

قسمتهای مقدار دهی اولیه و گام حرکت

می‌توانند شامل چندین دستور باشند که با

،

از یکدیگر جدا می‌شوند.

amirjalili.ir

برنامه‌سازی پیشرفته

مثال:

```
for ( i = 0 , sum = 0 ; i < 20 ; i ++ , j -- )
{
    sum = sum + i + j ;
    cout<<sum ;
}
```

برنامه‌سازی پیشرفته

مثال:

```
for(i=10,j=0;(i>0)&&(j<=20);i -- , j ++)
    cout<<i + j;
```


برنامه بنویسید ۵ عدد را گرفته و میانگین را حساب کند

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i,avg,x,sum=0;
```

ادامه مثال

```
for(i=1;i<=5;i++)
{
cout<<"please enter number";
cin>>x;
sum+=x;
}
avg=sum/5;
Cout<<"average of five number is:”<<avg;
getch(); }
```


برنامه ای که عددی را دریافت و فاکتوریل آن را حساب می کند

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i,x,fact=1;
Cout<<"please enter x";
Cin>>x;
```

ادامه مثال

```
for(i=1;i<=x;i++)
{
fact*=i;
}
Cout<<x<<"!="<<fact;
getch();
}
```


ادامه مثال:روش دوم

```
for(i=x ; i>=1 ; i--)
{
  fact*=i;
}
```

برنامه ای بنویسید که عددی را از کاربر دریافت کند نشان دهد عدد کامل است یا نه؟


```
#include<stdio.h>
#include<conio.h>
void main()
{
int  i,n,sum=0;
cin>>n;
```


ادامه مثال


```
for(i=1;i<=n/2;i++)
  if((n%i)==0)
     sum+=i;
if(n==sum)
cout<<"kamel";
getch();
}
```

برنامه‌سازی پیشرفته

حلقهٔ while

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


amirjalili.ir

برنامه‌سازی پیشرفته

مثال:

```
int  sum = 0 , i = 0 ;
while  ( i != -1 )
{
      sum = sum + i ;
      cin>>i;
}
```

برنامه ای بنویسید یک عدد صحیح و مثبت را از کاربر دریافت کند مجموع ارقام عدد را برگرداند

```
#include<stdio.h>
#include<conio.h>
void main()
{
int n,sum=0,s;
cout<<"please enter a number";
cin>>n;
while(n>0)
```


ادامه مثال

```
{
s=n%10;
sum+=s;
n/=10;
}
cout<<sum;
getch();
}
```

برنامه ای که تعدادی عدد مثبت را بگیرد و ماکزیمم آن را حساب کند

```
#include<staio.h>
#include<conio.h>
void main()
{
int x, max,counter=0;
cout<<"please enter first number";
cin>>x;
max=x;
while(x>0)
```


ادامه مثال

```
{
cout<<"enter next number";
cin>>x;
counter++;
if(max<x)
max=x;
}
Cout<<"maximum of"<< counter<<" number=" <<max;
getch();
}
```

برنامه‌سازی پیشرفته

حلقهٔ do ... while

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


تفاوت while با do-while

- حلقه while ممکن است اصلاً تکرار نگردد. (زمانیکه شرط همان لحظه اول غلط باشد)
- ولی do-while حداقل یکبار تکرار میگردد.

برنامه‌سازی پیشرفته

مثال:

```
int n=0,sum=0;
    do {
        sum = sum + n ;
        cin>>n;
      } while (n != -1);
```


برنامه‌سازی پیشرفته

تبدیل حلقهٔ while به for

```
for ( ; شرط حلقهٔ while ; )
{
    بدنهٔ حلقهٔ while
}
```

برنامه‌سازی پیشرفته

تبدیل حلقهٔ while به for (روش دیگر)

for (; شرط حلقهٔ while ; بدنهٔ حلقهٔ while)
;

$$\sum_{i=1}^{n} i^2 = 1+4+9+...+n^2$$

```
int i,s=0,x;
cin>>n;
for(i=1;i<=n;i++)
{
x=i*i;
s+=x;
}
cout<<s;
```

$$\sum_{i=1}^{n}(-1)^i * (2)^i = -2+4-8+16+...+(-1)^n * (2)^n$$


```
int p=1,t=1,s=0;
for(i=1;i<=n;i++)
{
p*=-1;
t*=2;
s+=(p*t);
}
```


$$\sum_{i=1}^{n}(i+1)! = 2+6+24+...+(n+1)!$$


```
int f=1,s=0;
for(i=2;i<=n+1;i++)
{
f*=i;
s+=f;
}
```

برنامه‌سازی پیشرفته

جلسهٔ ششم

آرایه

مجموعه‌ای از داده‌های:

همنوع و همنام

ترتیب‌دار توسط اندیسها

پشت سر هم

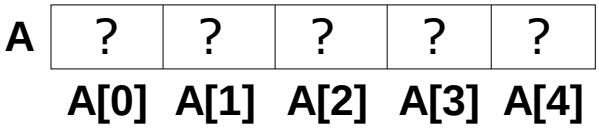
با حداکثر عناصر مشخص

برنامه‌سازی پیشرفته

آرایهٔ یک بعدی

نوع عناصر آرایه نام آرایه [تعداد عناصر آرایه];

```
int  A [ 5 ] ;
```



amirjalili.ir

برنامه‌سازی پیشرفته

چگونگی ارجاع به عناصر آرایه

نام آرایه [اندیس عنصر مورد نظر];

مثال:

```
A [ 3 ] = 124 ;
```

برنامه‌سازی پیشرفته

مثال: خواندن عناصر یک آرایه از ورودی

```
int  a [ 100 ] , i  ;
for (i = 0 ; i < 100 ; i ++)
     cin>>a[ i ];
```


amirjalili.ir

برنامه‌سازی پیشرفته

آرایه‌های چندبعدی

نوع عناصر آرایه نام آرایه [بعد ۱] [بعد ۲] ... [بعد n];

```
int   A [ 10 ] [ 12 ] [ 20 ] ;
```

دستیابی به عناصر آرایه:

```
A [0] [1] [2] = 400 ;
```

برنامه‌سازی پیشرفته

مثال: مقداردهی اولیه به عناصر یک آرایهٔ دو بعدی

```
int  a [10] [20] , row , col ;
for (row = 0 ; row < 10 ; row ++)
    for (col = 0 ; col < 20 ; col ++)
      a [row] [col]  = 0 ;
```


برنامه ای که یک آرایه ده خانه ای را دریافت کند و مجموع عناصر آنرا نشان دهد.

```
void main()
{
int i,s=0,a[10];
for(i=0;i<10;i++)
  cin>>a[i];
for(i=0;i<10;i++)
  s+=a[i];
cout<<"hasel="<<s;
getch();
}
```

برنامه ای که یک آرایه ده خانه ای را دریافت کند و تعداد عناصر فرد آنرا نشان دهد.

```
void main()
{
int i,counter=0,a[10];
for(i=0;i<10;i++)
{
  cin>>a[i];
  if(a[i]%2==1)
    counter++;
}
cout<<counter;
}
```


برنامه‌سازی پیشرفته

رشته

آرایه‌ای از کاراکتر

char نام رشته [حداکثر طول رشته];

char Str [20] ;

برنامه‌سازی پیشرفته

مقداردهی اولیه و نحوهٔ ذخیره

```
char  S [5] = “ali” ;
```

S	a	l	i	\0	?
	S[0]	S[1]	S[2]	S[3]	S[4]

amirjalili.ir

برنامه‌سازی پیشرفته

ورودی – خروجی رشته‌ها

```
scanf (“%s” , str);

gets (str);   یا  cin>>str;

printf (“%s” , str);

puts (str);    یا cout<<str;
```

برنامه‌سازی پیشرفته

فرق scanf و gets

در **scanf** کاراکتر فضای خالی مرز یا پایان رشته است

در حالی که

در **gets** فقط **Enter** پایان رشته است

مثال: رشتهٔ "Computer Science"

amirjalili.ir

برنامه‌سازی پیشرفته

نکته

برای کار با رشته‌ها نمی‌توان از اپراتورها استفاده کرد

بلکه باید

از توابع مربوط به رشته‌ها استفاده کرد.

<string.h>

برنامه‌سازی پیشرفته

مثال

```
char  s [10] ;
s = “ali” ;                  // نادرست
if ( s == “ABC”)             // نادرست
printf (“%s” , s) ;
```


amirjalili.ir

برنامه‌سازی پیشرفته

مهمترین توابع رشته‌ای

طول رشته: **strlen (s)**

انتساب: **strcpy (s1 , s2)**

مقایسه: **strcmp (s1 , s2)**

برنامه‌سازی پیشرفته

مثال برنامه‌نویسی

برنامه‌ای بنویسید که تعدادی نام از ورودی دریافت کرده و بمحض دریافت نام **ali** تعداد نامهای دریافتی را چاپ کند.

برنامه‌سازی پیشرفته

```
#include  <stdio.h>
#include  <string.h>
void  main (){
   char  s [30] ;
   int  count=0 ;
```

برنامه‌سازی پیشرفته

```
gets (s);
while( strcmp (s , “ali”) != 0  )
{
   count ++ ;
   gets (s) ;
}
cout<<“The number is:”<<count ;
}
```


برنامه‌سازی پیشرفته

پیمایش رشته

```
char s[100];
int  i=0;
gets(s);
while(s[i] != NULL)
{
بلوک پیمایش
i ++;
}
```

برنامه ای بنویسید رشته ای را دریافت بدون استفاده از توابع رشته طول آن را حساب کند؟

```
#include<stdio.h>
#include<conio.h>
void main()
{
char s[100];
int  counter=0;
cout<<"please  enter string";
gets(s);
```



```
while(s[counter]!=NULL)
{
counter++;
}
cout<<counter;
getch();
}
```

محاسبه تعداد کاراکتر A یا a در رشته ورودی

```
void main()
{
  char a[100];
  int  i=0,counter=0;
  gets(a);
  while  (a[i]!=NULL)
   {
     if((a[i]==‘A’)||(a[i]==‘a’))
       counter++;
     i++;
   }
cout<<counter;
}
```


تبدیل کاراکتر A یا a در رشته ورودی به کاراکتر *

```
void main()
{
  char a[100];
  int  i=0;
  gets(a);
  while  (a[i]!=NULL)
   {
     if((a[i]==‘A’)||(a[i]==‘a’))
       a[i]=‘*’;
     i++;
   }
  cout<<a;
getch();
}
```

برنامه‌سازی پیشرفته

جلسهٔ هفتم

amirjalili.ir

برنامه‌سازی پیشرفته

توابع

```
( لیست پارامترها )    نام تابع    نوع خروجی تابع
{
        دستورات
        ...
}
```

برنامه‌سازی پیشرفته

```
int  factorial ( int n )
{
   int i , f = 1 ;
   for ( i = 1 ; i <= n ; i ++ )
            f = f * i ;
   return f ;
}
```


amirjalili.ir

برنامه‌سازی پیشرفته

فراخوانی تابع

```
int  fact ;
fact = factorial ( 12 ) ;
```

در صورتی که پارامتری نداشته باشد ذکر () الزامی است.

```
ret = f () ;
```

برنامه‌سازی پیشرفته

برای تابعی که خروجی ندارد از کلمهٔ **void** استفاده می‌شود.

```
void  f ( int i , float j )
{
...
}
```


تابعی بنویسید که عددی را از کاربر دریافت کند و اول بودن آن را بررسی کند؟

```
void prim (int n)
{
int i=2
While(i<=n/2)
{
if(n%i==0)
{
```

```
cout<<“no”;
i=n;
}
else
i++;
}
if(i!=n)
cout<<"ok";
}
```


برنامه‌سازی پیشرفته

در برنامه همهٔ توابع در یک سطح هستند به این معنی که در داخل یک تابع نمی‌توان تابع دیگری تعریف کرد.

برنامه‌سازی پیشرفته

نکاتی در مورد نوشتن توابع

- بدون پرداختن به جزئیات پیاده‌سازی، پارامترها و خروجی را طراحی کنید.
- تابع باید فقط به آنچه نیاز دارد دسترسی داشته باشد **(Information hiding)**
- برای ارتباط با تابع از پارامترها استفاده کنید.

برنامه‌سازی پیشرفته

آرایهٔ یک بعدی بعنوان پارامتر تابع

```
void  f ( int x [ ] )
{  ...  }
void  main () {
int  a [10] ;  ...
f (a) ;
}
```

برنامه‌سازی پیشرفته

پارامترهای تابع

Call by value :

تغییر پارامتر در داخل تابع تاثیری بر فراخواننده ندارد.

Call by reference :

تغییر پارامتر در داخل تابع بر فراخواننده تاثیر دارد.

amirjalili.ir

برنامه‌سازی پیشرفته

شکل کلی یک برنامهٔ C

برنامه‌سازی پیشرفته

include section

Global Variables

تابع ۱
⋮
تابع n
تابع main

برنامه‌سازی پیشرفته

مقایسه تفاوتهای:

۱- تابع

۲- برنامه

۳- برنامه با استفاده از تابع

برنامه‌سازی پیشرفته

۱- تابع:(به تنهایی قابل اجرا نیست)

```
int  factorial ( int n )
{
   int i , f = 1 ;
   for ( i = 1 ; i <= n ; i ++ )
            f = f * i ;
   return f ;
}
```


amirjalili.ir

برنامه‌سازی پیشرفته

۲- برنامه

```
void main()
{
int  i , n, f = 1 ;
cin>>n;
for ( i = 1 ; i <= n ; i ++ )
        f = f * i ;
cout<<f;
getch();
}
```

برنامه‌سازی پیشرفته

۳- برنامه به کمک تابع:

```
int  factorial ( int n )
{
      int i , f = 1 ;
      for ( i = 1 ; i <= n ; i ++ )
                        f = f * i ;
      return f ;
}
void main()
{
int  n ;
cin>>n;
cout<<factorial(n);
getch();
}
```


amirjalili.ir

برنامه‌سازی پیشرفته

متغیرها

محدودهٔ شناسایی متغیر (Scope)

طول عمر متغیر (Life time)

برنامه‌سازی پیشرفته

انواع متغیرها

1. عمومی: خارج از توابع تعریف می‌شوند.

محدوده: از محل تعریف تا انتهای برنامه

طول عمر: از شروع اجرای برنامه تا پایان آن

amirjalili.ir

برنامه‌سازی پیشرفته

انواع متغیرها

2. محلی: در داخل یک تابع تعریف می‌شوند.

محدوده: در داخل تابعی که تعریف شده‌اند.

طول عمر: با شروع اجرای تابع، ایجاد و با پایان اجرای آن از بین می‌روند.

برنامه‌سازی پیشرفته

مسالهٔ همنام بودن متغیرها

نزدیکترین تعریف در نظر گرفته می‌شود.

(ارجحیت تعریف محلی به عمومی)

مثال:

```
#incude  <stdio.h>
int  i , j ;
int  f1 ( int j )
{
    j = j + i ;
    return j * j ;
}
int  k ;
void  f2 ( void )
{
    int n ;
    scanf (“%d” , &n);
    j = f1 (n)  + k ;
}
```

برنامه‌سازی پیشرفته

```
void main ()
{
   int i ;
   for (i = 0 , k = 10 ; i < 10 ; i ++ )
   {
       f2 () ;
      printf (“j=%d” , j) ;
   }
}
```


برنامه‌سازی پیشرفته

متغیرهای استاتیک محلی

static نوع متغیر نام متغیر ;

static int s ;

برنامه‌سازی پیشرفته

متغیرهای استاتیک محلی

فقط یک بار مقدار اولیه می‌گیرند.

```
static  int  s = 0 ;
```


برنامه‌سازی پیشرفته

توابع بازگشتی

1. تابع خودش، خودش را فراخوانی میکند.
2. حتماً یک شرط برای پایان فراخوانی ها وجود دارد.

برنامه‌سازی پیشرفته

تابع بازگشتی برای فاکتوریل

```
int  fact2 ( int n)
{
if (n<=1)
  return  1;
else
  return  n*fact2(n-1);
}
```


amirjalili.ir

برنامه‌سازی پیشرفته

تابع بازگشتی برای توان

```
int  pow2 ( int n, int m)
{
if (m<1)
  return  1;
else
  return  n*pow2(n,m-1);
}
```

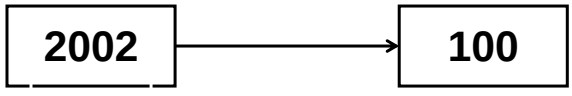
برنامه‌سازی پیشرفته

جلسهٔ هشتم

برنامه‌سازی پیشرفته

اشاره‌گر (Pointer)

اشاره‌گر متغیری است که حاوی آدرس یک متغیر است و در واقع به آن اشاره می‌کند.



برنامه‌سازی پیشرفته

لزوم استفاده از اشاره‌گرها در C

- درک و استفادهٔ بهتر از آرایه‌ها و رشته‌ها
- استفاده از پارامترهای Call by reference در توابع
- تخصیص حافظهٔ پویا
- ایجاد و کار با ساختمانهای داده‌ای پیچیده

برنامه‌سازی پیشرفته

amirjalili.ir

نحوهٔ تعریف متغیر اشاره‌گر

نام متغیر اشاره‌گر * نوع داده‌ای که به آن اشاره می‌کند;

```
int   * p ;
char        * pc ;
float    * fp ;
```

برنامه‌سازی پیشرفته

عملگرهای اشاره‌گر

& : آدرس عملوند خود را مشخص می‌کند.

& i عملوند آن نام یک متغیر است.

***** : محتوای عملوند خود را مشخص می‌کند.

*** p** عملوند آن نام یک متغیر اشاره‌گر است.

amirjalili.ir

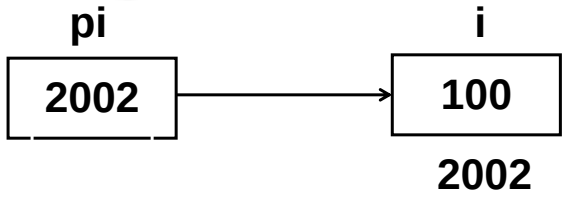
برنامه‌سازی پیشرفته

مثال

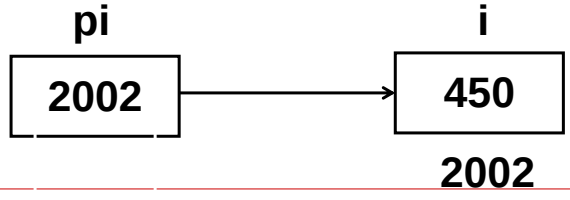
```
int  i = 100 ;
int  * pi ;
```

برنامه‌سازی پیشرفته

pi = & i ;



* pi = 450 ;



amirjalili.ir

برنامه‌سازی پیشرفته

عملگرهای مجاز دیگر

- انتساب اشاره‌گرها به یکدیگر
- جمع و تفریق با یک ثابت یا عبارت محاسباتی
- عملگرهای رابطه‌ای

برنامه‌سازی پیشرفته

مثال

```
int  * p1 , * p2 , i ;
p1 = p2 ;
p1 = p1 + 2 ;
p1 = p1 + i * 2 ;
p1 == p2
p2 <= p1
```


amirjalili.ir

برنامه‌سازی پیشرفته

اشاره‌گرها و توابع

پارامتر **Call by reference** پارامتری است که تغییرات آن در داخل تابع عینا به فراخوانندهٔ آن منعکس می‌شود.

برای استفاده از این نوع پارامتر باید از اشاره‌گر استفاده کرد.

برنامه‌سازی پیشرفته

مثال برنامه‌نویسی

تابعی که محتوای دو متغیر را با یکدیگر عوض می‌کند.

amirjalili.ir

برنامه‌سازی پیشرفته

روش غلط

```
void  swap (int  a , int  b)
{
   int  temp ;
   temp = a ;
   a = b ;
   b = temp ;
}
```

برنامه‌سازی پیشرفته

روش صحیح

```
void  swap (int  *a , int  *b)
{
   int  temp ;
   temp = *a ;
   *a = *b ;
   *b = temp ;
}
```


برنامه‌سازی پیشرفته

```
void  main ()
{
   int x = 10 , y = 20 ;
   swap ( &x , &y ) ;
   cout<<“x =“<<x “, y = ” << y ;
}
```

خروجی:

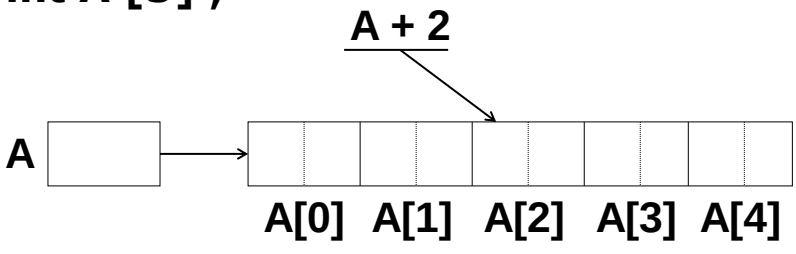
```
x = 20 , y = 10
```

برنامه‌سازی پیشرفته

اشاره‌گر و آرایه

نام آرایه اشاره‌گر به اولین عنصر آن است.

```
int A [5] ;
```



amirjalili.ir

برنامه‌سازی پیشرفته

بنابراین

A [2]

معادل

*** (A + 2)**

```
A [2] = 100 ;
* (A + 2) = 100 ;
```

برنامه‌سازی پیشرفته

رشته و اشاره‌گر

رشته نیز یک آرایه است

و بنابراین

نام رشته اشاره‌گر به اولین عنصر آن است.

```
scanf (“%s” , s) ;
```


amirjalili.ir

برنامه‌سازی پیشرفته

آرایه و رشته بعنوان پارامتر تابع

```
void  f (int  *a , char *s) {
   …
   a [2] = 100 ;
   strcpy (s , “Ali”) ;
}
```

برنامه‌سازی پیشرفته

```
void  main () {
   int  b [10] ;
   char  str [20] ;
   …
   f (b , str) ;
}
```


برنامه‌سازی پیشرفته

تخصیص حافظهٔ پویا

؟

نیاز به حافظه‌ای که مقدار آن موقع نوشتن برنامه مشخص نیست بلکه در زمان اجرا مشخص می‌شود.

برنامه‌سازی پیشرفته

تابع تخصیص حافظهٔ پویا

void * malloc (اندازهٔ حافظهٔ مورد نیاز به بایت)

```
int  *p ;
p = (int *) malloc ( sizeof (int) ) ;
```


amirjalili.ir

برنامه‌سازی پیشرفته

تعریف یک آرایه بصورت پویا

```
int  *A ;
A = (int *) malloc ( sizeof (int) * 100 ) ;
```

با دستورات فوق آرایه‌ای با ظرفیت ۱۰۰ عنصر با نام **A** و بصورت پویا ایجاد می‌شود. `int A[100];`

برنامه‌سازی پیشرفته

تابع آزادسازی حافظهٔ پویا

free (اشاره‌گری که قبلا به آن حافظه اختصاص داده شده)

```
int  *A ;
A = (int *) malloc ( sizeof (int) * 100 ) ;
//  ...کار با حافظهٔ پویا
free (A) ;
```


برنامه‌سازی پیشرفته

مثال

برنامه‌ای که نمرهٔ تعدادی دانشجو را دریافت کرده و ...

تعداد دانشجویان ؟

برنامه‌سازی پیشرفته

روش غلط

```
int  n ;
float  A [n] ;
...
cin>>n;
```


برنامه‌سازی پیشرفته

روش صحیح

```
int  n ;
float  *A ;
...
cin>>n;
A = (float *) malloc (sizeof (float) * n) ;
کار با آرایهA  ...  //
free (A) ;
```

برنامه‌سازی پیشرفته

جلسهٔ نهم

برنامه‌سازی پیشرفته

ساختمان (Structure)

برای نگهداری اطلاعات
از انواع داده‌ای مختلف (بر خلاف آرایه)
مرتبط با یکدیگر
تحت یک نام

برنامه‌سازی پیشرفته

```
struct  نام ساختمان
{
    نام فیلد ۱  نوع فیلد ۱ ;
    ...
   نام فیلد n  نوع فیلد n ;
} اسامی متغیرهای از نوع ساختمان ;
```

قسمت متغیرها در آخر اختیاری است و بدون آن فقط یک نوع ساختمان تعریف می‌شود.

amirjalili.ir

برنامه‌سازی پیشرفته

مثال: ساختمانی برای نگهداری اطلاعات دانشجو

```
struct  student {
   int  student_no ;
   char  fname [20] ;
   char  lname [10] ;
   float  average ;
} ;
struct  student stud1 , stud2 ;
```

برنامه‌سازی پیشرفته

دستیابی به فیلدهای ساختمان

نام فیلد . نام متغیر از نوع ساختمان

```
stud1.average = 17.3 ;
scanf (“%d” , &stud1. student_no) ;
strcpy (stud1.lname , “Rezaie”) ;
```


amirjalili.ir

برنامه‌سازی پیشرفته

نکته

انتساب ساختمانهای همنوع به یکدیگر امکان‌پذیر است که با این عمل تک‌تک فیلدها منتقل خواهد شد.

```
struct  student  stud1 , stud2 ;
...
stud1 = stud2 ;
```

برنامه‌سازی پیشرفته

آرایه‌ای از ساختمانها

آرایه‌ای برای نگهداری اطلاعات دانشجویان یک کلاس مثال:

```
struct  student  A [100] ;
A [15].average = 12.25 ;
strcpy (str1 , A [2].fname) ;
```


برنامه‌سازی پیشرفته

ساختمان و اشاره‌گر

دسترسی به فیلدها توسط اشاره‌گر به ساختمان

نام فیلد <- نام اشاره‌گر به ساختمان

```
struct  student  *p ;
p = &stud1 ;
p-> average = 14.5 ;
```

برنامه‌سازی پیشرفته

مثال برنامه‌نویسی: جابجایی محتوای دو رکورد

```
void  swap (struct student *s1 , struct student *s2)
 {
   struct student temp ;
   temp = *s1 ;
   *s1 = *s2 ;
   *s2 = temp ;
 }
```


برنامه‌سازی پیشرفته

union

محلی از حافظه است که توسط دو یا چند متغیر بطور مشترک استفاده می‌شود

البته

نه بصورت همزمان

برنامه‌سازی پیشرفته

تعریف union

```
union  نام نوع یونیون
{
   نام فیلد ۱  نوع فیلد ۱ ;
   ...
   نام فیلد n  نوع فیلد n ;
} اسامی متغیرهای از نوع یونیون ;
```


amirjalili.ir

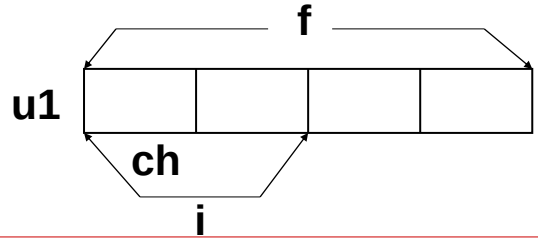
برنامه‌سازی پیشرفته

مثال

```
union  u
{
   int i ;
   char ch ;
   float f ;
} u1 ;
```

برنامه‌سازی پیشرفته

حافظه‌ای که در نظر گرفته می‌شود
به اندازهٔ طول عنصری است که
بیشترین طول را دارد.



amirjalili.ir

برنامه‌سازی پیشرفته

جلسهٔ دهم

برنامه‌سازی پیشرفته

فایل

برای انتقال خروجی برنامه‌ها به حافظهٔ پایدار

بدلیل

ماندگاری آنها

ایجاد ارتباط بین برنامه‌ها (فایل بعنوان ورودی)

amirjalili.ir

برنامه‌سازی پیشرفته

انواع فایلها

متن (Text)

باینری (Binary)

برنامه‌سازی پیشرفته

ساختار اطلاعات در فایل متن

رشته‌ای از کاراکترها

برای مثال عدد ۲۵۳ بصورت سه کاراکتر ۲، ۵ و ۳ ذخیره شده و سه بایت را اشغال می‌کند.

amirjalili.ir

برنامه‌سازی پیشرفته

هر سطر به کاراکترهای پایان سطر ختم (CR/LF) ختم می‌شود.

پایان فایل را کاراکتری با کد اسکی ۲۶ مشخص می‌کند.

برنامه‌سازی پیشرفته

مثال

Adgvvh12<CR/LF>125534
6asd6633<CR/LF><\26>

برنامه‌سازی پیشرفته

ساختار اطلاعات در فایل باینری

داده‌ها به همان شکل موجود در حافظه ذخیره می‌شوند.

برای مثال عدد ۲۵۳ چون یک عدد صحیح است در ۲ بایت ذخیره می‌شود.

برنامه‌سازی پیشرفته

پایان سطر در فایل باینری مفهومی ندارد.

پایان فایل از طول آن مشخص می‌شود
(دیگر کاراکتر ۲۶ پایان دهنده نیست)

amirjalili.ir

برنامه‌سازی پیشرفته

مثال

♣:↓☻↕●¶¤¥£Θ€A‡ÿèâ:↓☻↕
¥£Θ€A‡ÿèâ

داده‌هایی با نمایش قابل فهم نیستند

برنامه‌سازی پیشرفته

امکانات منطقی لازم دراستفاده از فایلها

- تعریف متغیر از نوع فایل
- تعیین محل و عنوان فیزیکی فایل
- باز کردن فایل
- خواندن داده از فایل ورودی

برنامه‌سازی پیشرفته

amirjalili.ir

- نوشتن داده در فایل خروجی
- اضافه کردن داده به انتهای فایل
- آزمون انتهای فایل درهنگام خواندن
- آزمون انتهای سطر درهنگام خواندن (فقط فایل متن)
- قرار دادن <CR/LF> در انتهاي سطر (فقط فایل متن)
- بستن فایل

برنامه‌سازی پیشرفته

باز کردن فایل

FILE * fopen(char *filename, char *mode)

خروجی تابع به یک متغیر اشاره‌گرفایل نسبت داده می‌شود.

filename: مسیر و نام فایل روی دیسک

mode: مشخص کنندهٔ چگونگی باز شدن فایل

amirjalili.ir

برنامه‌سازی پیشرفته

mode ترکیبی از کارکترهای:

r فایل ورودی

w فایل خروجی

a اضافه کردن داده به انتهای فایل

t فایل متن

b فایل باینری

+ فایل ورودی و خروجی

برنامه‌سازی پیشرفته

مثال

rt : فایل متن به عنوان ورودی

at : فایل متن برای اضافه کردن داده به انتهای فایل

wb : فایل باینری به عنوان خروجی

r+b : فایل باینری به عنوان ورودی و خروجی

برنامه‌سازی پیشرفته

مثال

```
FILE  *fp ;
fp = fopen (“c:\test.txt” , “wt”) ;
if (fp == NULL)
      printf (“can not open file.”) ;
```

همیشه باید برای اطمینان، باز شدن موفقیت آمیز فایل را تست کرد.

برنامه‌سازی پیشرفته

بستن فایل

در انتهای کار با فایل آن را می‌بندیم.

```
fclose (FILE *fp) ;
```

مثال:

```
fclose (fp) ;
```


amirjalili.ir

برنامه‌سازی پیشرفته

مهمترین توابع ورودی - خروجی فایل

fscanf

fprintf

fread

fwrite

برنامه‌سازی پیشرفته

fprintf (FILE *fp , " عبارت ۱ " , عبارت ۲);

fscanf (FILE *fp , " عبارت ۱ " , عبارت ۲);

دقیقا همانند printf و scanf با تفاوت

ذکر اشاره‌گر فایل در ابتدای آنها

amirjalili.ir

برنامه‌سازی پیشرفته

مثال

```
FILE *fp1 , *fp2 ;
fp1 = fopen ("c:\test.txt" , "rt") ;
fp2 = fopen ("c:\ali.dat" , "wb") ;
...
fscanf (fp1 , "%d %f %s" , &i , &f , str) ;
fprintf (fp2 , "%f, %s" , f , str) ;
```

برنامه‌سازی پیشرفته

fwrite و fread

بیشتر برای ورودی – خروجی رکورد استفاده می‌شود

ولی

در همهٔ موارد می‌تواند بکار رود

و

سرعت بالایی نیز دارد.

amirjalili.ir

برنامه‌سازی پیشرفته

```
fread (void *buffer, int num_byte ,
       int count, FILE *fp)
```

buffer : محلی از حافظه که داده‌های خوانده شده باید در آن قرار گیرند.

num_byte : طول داده‌ای که باید خوانده شود.

count : تعداد عناصری (به طول **num_byte**) که باید از فایل خوانده شوند.

fp : اشاره‌گر به فایلی که باید از آن خوانده شود.

برنامه‌سازی پیشرفته

مثال

```
struct  student  stud ;
fread ( &stud , sizeof (struct student)
   ,1 , fp ) ;
```


amirjalili.ir

برنامه‌سازی پیشرفته

مثال

```
int  a [20] ;
fread (a , sizeof (int) , 20 , fp ) ;
```

دریافت ۲۰ عدد از فایل و قرار دادن آن در آرایه

برنامه‌سازی پیشرفته

fwrite (void *buffer, int num_byte , int count, FILE *fp)

پارامترهای این دستور دقیقا مانند **fread** است

با این تفاوت که

buffer محلی از حافظه است که داده‌هایی که باید در فایل نوشته شوند در آن قرار می‌گیرند.

amirjalili.ir

برنامه‌سازی پیشرفته

تشخیص پایان فایل

```
int  feof (FILE *fp)
```

مثال:

```
if ( feof (fp) ) {
    printf (“End of File”) ;
    fclose (fp) ;
{
```

برنامه‌سازی پیشرفته

بازگشت به ابتدای فایل

```
rewind (FILE  *fp)
```

مثال:

```
rewind (fp) ;
```


amirjalili.ir

برنامه‌سازی پیشرفته

دسترسی تصادفی به فایل

```
fseek (FILE *fp, long num_byte, int origin)
```

این تابع **Cursor** فایل را به تعداد **num_byte** بایت از محل **origin** تغییر مکان می‌دهد.

برنامه‌سازی پیشرفته

origin :

- SEEK_SET : ابتدای فایل
- SEEK_CUR : موقعیت فعلی فایل
- SEEK_END : انتهای فایل

برنامه‌سازی پیشرفته

amirjalili.ir

مثال

```
fseek (fp , 10 , SEEK_SET);
```

Cursor فایل را از اول فایل ۱۰ بایت جلو می‌برد.

```
fseek (fp , -20 , SEEK_END);
```

Cursor فایل را از انتهای فایل ۲۰ بایت عقب می‌برد.

برنامه‌سازی پیشرفته

جلسهٔ یازدهم

برنامه‌سازی پیشرفته

amirjalili.ir

ايده شيءگرايي

- شئ گرايي يا OO(Object Oriented) يك روش جديد برنامه نويسي مي‌باشد كه در آن از ويژگي ساختيافته براي اشياء همراه با سه ويژگي قوي و جديد زيراستفاده مي‌شود:

1. چند ريختي
2. وراثت
3. كپسوله سازي

برنامه‌سازی پیشرفته

برنامه نويسي شيءگرا

- به برنامه نويسي شيءگرا OOP يا (Object Oriented Programing) گويند.
- زبان برنامه نويسي C++ امكان استفاده از OOP را به راحتي فراهم مي‌نمايد.

amirjalili.ir

برنامه‌سازی پیشرفته

كلاس و اشياء

- اشيا را مي‌توان با توجه به مشخصات ورفتار آن‌ها دسته بندي كرد به اين دسته‌ها «كلاس» مي‌گويند و به نمونه‌هاي هر كلاس «شي» گفته مي‌شود.
- مشخصات هر شي را «صفت» مي‌نامند و به رفتارهاي هر شي «متد» مي‌گويند.

برنامه‌سازی پیشرفته

چندريختي

- به معناي يک چيز بودن و چند شکل داشتن است، چندريختي بيشتر در وراثت معنا پيدا مي‌کند.
- چند ريختي مشخصه‌اي است که به يك شيء امکان ميدهد که با تعدادي از سيستم ها يا عميات يا دستگاهها، مورد استفاده قرار گيرد.

amirjalili.ir

برنامه‌سازی پیشرفته

وراثت

- ارث بري فرآيندي است که بوسيله آن يك شي (object) مي‌تواند خاصيت‌هاي شي ديگري را دارا شود.
- يک شي وقتي متولد مي‌شود، خصوصيات و ويژگي‌هايي را از والد خود به همراه دارد .
- امتياز وراثت در اين است که علاوه بر کدهاي قبلي مي‌توان از کدهاي مشترک استفاده مي‌شود.

برنامه‌سازی پیشرفته

کپسوله سازي

- يعني اين كه داده‌هاي مرتبط، با هم تركيب شوند و جزييات پيادەسازي مخفي شود.
- مكانيزمي است كه code و data را بهم وصل نموده و هر دوي آنها را از استفاده‌هاي غيرمجاز مصون نگه مي‌دارد.

amirjalili.ir

برنامه‌سازی پیشرفته

اعلان كلاس

- اعلان كلاس با كلمۀ كليدي class شروع مي‌شود سپس نام كلاس مي‌آيد.
- هر عضوي كه ذيل عبارت public اعلان شود، يک «عضو عمومي» محسوب مي‌شود.
- هر عضوي كه ذيل عبارت private اعلان شود، يک «عضو خصوصي» محسوب مي‌شود.

برنامه‌سازی پیشرفته

مثال

```
class X
{
  private :
   int s [SIZE];
   int t;
  public:
   int a,b;
 };
```


برنامه‌سازی پیشرفته

سازنده

- وظيفهٔ تابع سازنده اختصاص حافظهٔ لازم براي شيء جديد و مقداردهي آن براي اجراي وظايفي كه در اين تابع منظور شده ميباشد.
- هر كلاس مي‌تواند چندين سازنده داشته باشد. در حقيقت تابع سازنده مي‌تواند چندشكلي داشته باشد.

برنامه‌سازی پیشرفته

مثال

```
class X
{
   private :
    x() {a=0;b=1;}
   public:
    int a,b;
 };
```


برنامه‌سازی پیشرفته

نابود کننده

- سازنده وظیفه دارد تا منابع لازم را براي شي تخصيص دهد و نابودکننده وظیفه دارد آن منابع را آزاد کند.
- هر کلاس فقط یک نابودکننده دارد.
- اینکار با افزودن علامت ~ قبل از نام کلاس انجام میگیرد.

پایان