برنامه‌سازی پیشرفته

جلسهٔ اول

برنامه‌سازی پیشرفته

مقدمه و معرفی درس

برنامه‌سازی پیشرفته:

بیان مفاهیم پیشرفته برنامه‌سازی با استفاده از زبان C

برنامه‌سازی پیشرفته

منبع اصلی جهت مطالعه دانشجویان

کتاب: برنامه‌نویسی به زبان C

(ویرایش 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

۱. C یک زبان میانی است

برنامه‌سازی پیشرفته

سطوح زبانهای برنامه‌سازی

زبانهای سطح پایین

Assembly

زبانهای میانی

Java، C

زبانهای سطح بالا

Pascal، Ada، Cobol، Basic

برنامه‌سازی پیشرفته

ویژگیهای بارز زبان C

۲. C یک زبان ساختیافته است.

۳. C زبان برنامه‌نویسی سیستم است.

۴. C یک زبان قابل حمل است.

۵. C زبانی قابل انعطاف و قدرتمند است.

برنامه‌سازی پیشرفته

کلیات زبان C

- حساس به حروف (Case Sensitive)

  INT و int

- کلمات کلیدی کم

  مثال: for ، if ، while

نکته: کلیه 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

برنامه‌سازی پیشرفته

کلیات زبان C

- توضیحات بین /* و */ یا بعد از //

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

برنامه‌سازی پیشرفته

برنامهٔ کامپیوتری

برنامه‌سازی پیشرفته

مجموعهٔ دستورات هر زبان برنامه‌نویسی

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

برنامه‌سازی پیشرفته

جلسهٔ دوم

برنامه‌سازی پیشرفته

انواع داده‌های اصلی

int

float

double

char

void

boolean ?!!

برنامه‌سازی پیشرفته

int

اعداد صحیح با دامنه محدود

برای کامپیوترهای شخصی دو بایت

-۳۲۷۶۷

+۲۷۶۲

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

برنامه‌سازی پیشرفته

انواع داده‌های 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

متغیرها

قوانین نامگذاری متغیرها:

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

برنامه‌سازی پیشرفته

متغیرها

اسامی مجاز:

count

c124

avg_grade

اسامی غیرمجاز:

1test

bin#tree

for

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 متغیر

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

برنامه‌سازی پیشرفته

تبدیل انواع

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

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

عملگرهای رابطه‌ای

> , >= , < , <=

== , !=

مثال:

x > y

x == y

x != y

برنامه‌سازی پیشرفته

عملگرهای منطقی

!

&&

||

مثال:

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

! (x > 20)

برنامه‌سازی پیشرفته

عملگرهای بیتی

(And) &

(Or) |

(Xor) ^

(Not) ~

(Right Shift) >>

(Left Shift) <<

برنامه‌سازی پیشرفته

مثال

```
char x = 7 , y ;

y = x & 2 ;

y = ~y ;
```

برنامه‌سازی پیشرفته

عملگرهای دیگر

- عملگرهای ترکیبی شامل:

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

که x = x + y معادل x += y

- غیره (در جای خود توضیح داده خواهند شد)

برنامه‌سازی پیشرفته

جلسهٔ سوم

برنامه‌سازی پیشرفته

ساختار یک برنامهٔ ساده

```
#include  < header file>
void    main()
{
   تعریف متغیرها
   دستورات اجرایی
}
```

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 : اطلاعاتی که باید به خروجی منتقل شوند.
(اختیاری است)

برنامه‌سازی پیشرفته

printf (“ عبارت ۱ ” , عبارت ۲)

عبارت ۱ می‌تواند شامل:

- اطلاعاتی 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
```

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

برنامه‌سازی پیشرفته

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:

۱. کاراکترهای فرمت. مشخص‌کنندهٔ نوع اطلاعات. مانند:

%c (کاراکتر)

%d (عدد صحیح)

برنامه‌سازی پیشرفته

۲. کاراکتر فضای خالی

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

۳. 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) ;
```

برنامه‌سازی پیشرفته

مثال برنامه‌نویسی

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

برنامه‌سازی پیشرفته

```
#include  <stdio.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) ;
}
```

برنامه‌سازی پیشرفته

جلسهٔ چهارم

برنامه‌سازی پیشرفته

دستورات کنترل روند برنامه

ساختارهای تصمیم

و

حلقه‌های تکرار

برنامه‌سازی پیشرفته

ساختارهای تصمیم

if

if else

switch

برنامه‌سازی پیشرفته

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

مثال:

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

برنامه‌سازی پیشرفته

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

برنامه‌سازی پیشرفته

مثال:

```
if  (x < y)
{
      x = x + y ;
      printf (“%d , %d”, x , y) ;
}
```

برنامه‌سازی پیشرفته

```
if (عبارت منطقی) {
        دستور ;
        ... ;
}
else {
        دستور ;
        ... ;
}
```

برنامه‌سازی پیشرفته

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

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>
void   main()  {
        int i ;
        scanf (“%d” , &i) ;
        if  (i % 2 == 0)
                printf (“The number is even.”) ;
        else   printf (“The number is odd.”) ;
}
```

برنامه‌سازی پیشرفته

دستورات شرطی متداخل

```
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 ;
```

برنامه‌سازی پیشرفته

دستور switch

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

برنامه‌سازی پیشرفته

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

برنامه‌سازی پیشرفته

.

.

.

default :

دستورات n

}

برنامه‌سازی پیشرفته

مثال:

```
char  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 ;
            printf (“Invalid operator.”) ;
}
```

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 بعدی و تا آخر اجرا خواهد شد.

برنامه‌سازی پیشرفته

```
int grade ;
switch  ( grade )  {
        case 18 :
        case 19 :
        case 20 :
                printf (“Good”) ;
                break ;
```

برنامه‌سازی پیشرفته

```
   case 10 :
                printf (“Acceptable”) ;
}
```

برنامه‌سازی پیشرفته

ساختارهای کنترل غیرشرطی

break

continue

goto

بدلیل پایین آوردن خوانایی

استفاده از آنها توصیه نمی‌شود.

برنامه‌سازی پیشرفته

جلسهٔ پنجم

برنامه‌سازی پیشرفته

ساختارهای تکرار

for

while

do ... while

برنامه‌سازی پیشرفته

حلقهٔ for

یکی از قویترین و کاملترین دستورات C

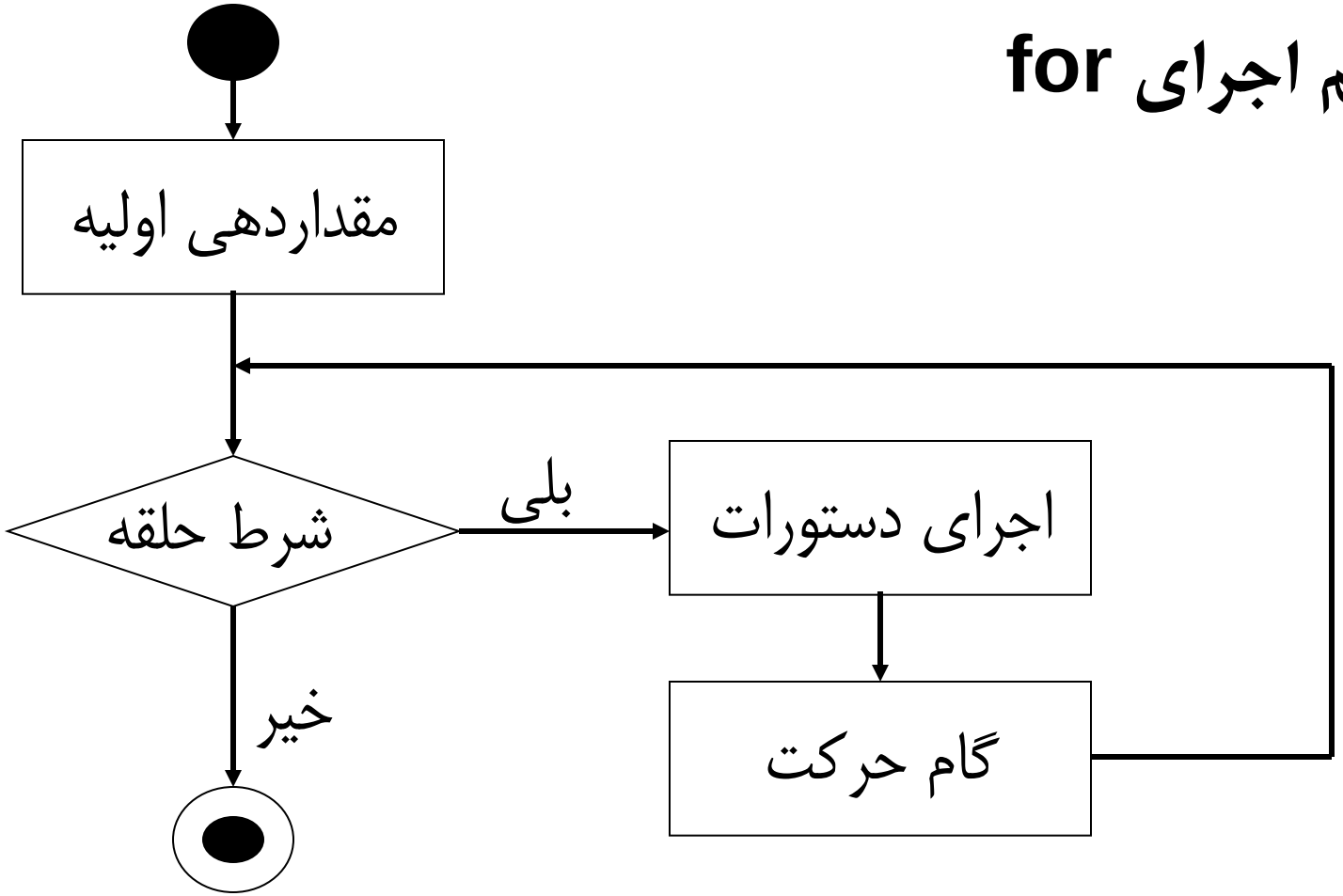
در تمامی کاربردهای حلقه می‌تواند بکار رود

برنامه‌سازی پیشرفته

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

برنامه‌سازی پیشرفته

الگوریتم اجرای for



برنامه‌سازی پیشرفته

مثال:

```
for ( i = 1 ; i < 10 ; i ++ )
{
    scanf (“%d” , &num) ;
    sum = sum + num ;
}
```

برنامه‌سازی پیشرفته

نکات

هر یک از ۳ قسمت فوق اختیاری هستند.

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

برنامه‌سازی پیشرفته

قسمتهای مقدار دهی اولیه و گام حرکت

می‌توانند شامل چندین دستور باشند که با

,

از یکدیگر جدا می‌شوند.

برنامه‌سازی پیشرفته

مثال:

```
for ( i = 0 , sum = 0 ; i < 20 ; i ++ , j -- )
{
    sum = sum + i + j ;
    printf (“%d” , sum) ;
}
```

برنامه‌سازی پیشرفته

مثال:

```
for  ( i = 10 , j = 0 ;
       (i > 0) && (j <=20) ;
       i -- , j ++)
           printf (“%d , “ , i + j) ;
```

برنامه‌سازی پیشرفته

حلقهٔ while

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

برنامه‌سازی پیشرفته

مثال:

```
int  sum = 0 , i = 0 ;
while  ( i != -1 )
{
        sum = sum + i ;
        scanf (“%d” , &i) ;
}
```

برنامه‌سازی پیشرفته

حلقهٔ do ... while

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

برنامه‌سازی پیشرفته

مثال:

```
do {
        sum = sum + n ;
        scanf (“%d” , &n) ;
}  while (n != -1)
```

برنامه‌سازی پیشرفته

مثال:

تبدیل حلقهٔ while به for

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

برنامه‌سازی پیشرفته

تبدیل حلقهٔ while به for (روش دیگر)

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

برنامه‌سازی پیشرفته

جلسهٔ ششم

برنامه‌سازی پیشرفته

توابع

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

برنامه‌سازی پیشرفته

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

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 )
...
```

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

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.

برنامه‌سازی پیشرفته

شکل کلی یک برنامهٔ C

برنامه‌سازی پیشرفته

include section

Global Variables

تابع ۱
⋮
تابع n

تابع main

برنامه‌سازی پیشرفته

متغیرها

محدودهٔ شناسایی متغیر (Scope)

طول عمر متغیر (Life time)

برنامه‌سازی پیشرفته

انواع متغیرها

۱. عمومی: خارج از توابع تعریف می‌شوند.

محدوده: از محل تعریف تا انتهای برنامه

طول عمر: از شروع اجرای برنامه تا پایان آن

برنامه‌سازی پیشرفته

انواع متغیرها

۲. 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)

برنامه‌سازی پیشرفته

مثال:

```
#incude  <stdio.h>
int  i , j ;
int  f1 ( int j )
{
   j = j + i ;
   return j * j ;
}
```

برنامه‌سازی پیشرفته

```
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

طول عمر: با اولین اجرای تابع ایجاد شده و تا پایان اجرای برنامه باقی می‌مانند.

در نتیجه:

با خروج از تابع مقدار خود را حفظ می‌کنند

برنامه‌سازی پیشرفته

متغیرهای استاتیک محلی

فقط یک بار مقدار اولیه می‌گیرند.

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

برنامه‌سازی پیشرفته

جلسهٔ هفتم

برنامه‌سازی پیشرفته

آرایه

مجموعه‌ای از داده‌های:

همنوع

ترتیب‌دار توسط اندیسها

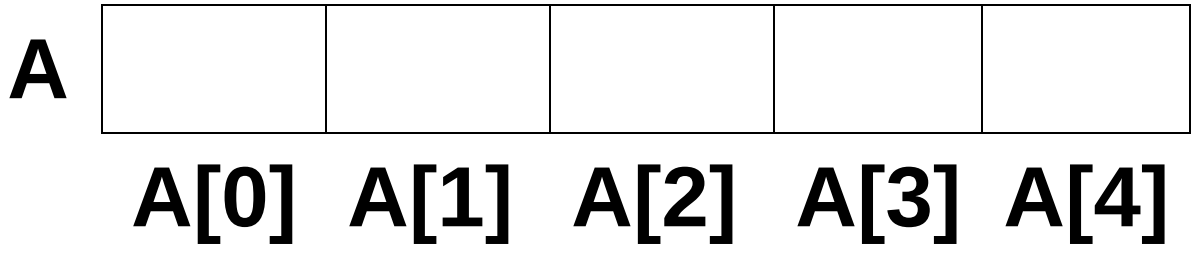
با حداکثر عناصر مشخص

برنامه‌سازی پیشرفته

آرایهٔ یک بعدی

[تعداد عناصر آرایه] نام آرایه نوع عناصر آرایه

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



برنامه‌سازی پیشرفته

چگونگی ارجاع به عناصر آرایه

[اندیس عنصر مورد نظر] نام آرایه

مثال:

A [3] = 124 ;

برنامه‌سازی پیشرفته

مثال: خواندن عناصر یک آرایه از ورودی

```
int  a [ 100 ] , i  ;

for (i = 0 ; i < 100 ; i ++)

     scanf (“%d”, &a[ i ] );
```

برنامه‌سازی پیشرفته

آرایهٔ یک بعدی بعنوان پارامتر تابع

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

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 ;
```

برنامه‌سازی پیشرفته

رشته

آرایه‌ای از کاراکتر

char نام رشته [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]

برنامه‌سازی پیشرفته

ورودی – خروجی رشته‌ها

scanf (“%s” , str)

gets (str)

printf (“%s” , str)

برنامه‌سازی پیشرفته

فرق scanf و gets

در scanf کاراکتر فضای خالی مرز رشته است

در حالی که

در gets فقط Enter پایان رشته است

مثال: رشتهٔ "Computer Science"

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) ;
```

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 ;
```

برنامه‌سازی پیشرفته

```
    gets (s);
    for ( count = 0 ; strcmp (s , “ali”) != 0 ; ){
        count ++ ;
        gets (s) ;
    } ;
    printf (“The number is: %d” , count) ;
}
```

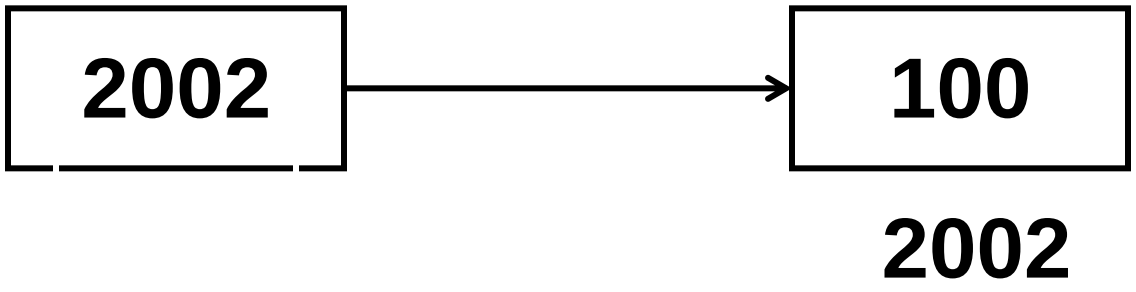
برنامه‌سازی پیشرفته

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

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.

عملوند آن نام یک متغیر است. `& i`

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

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

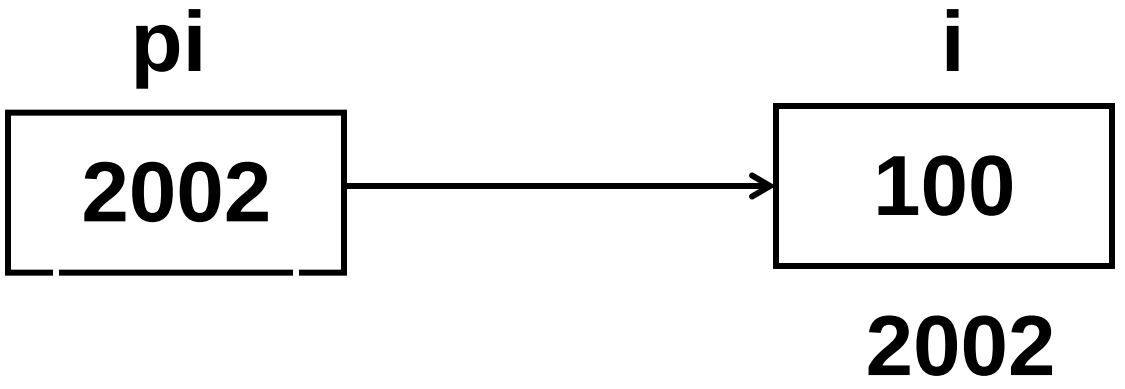
برنامه‌سازی پیشرفته

مثال

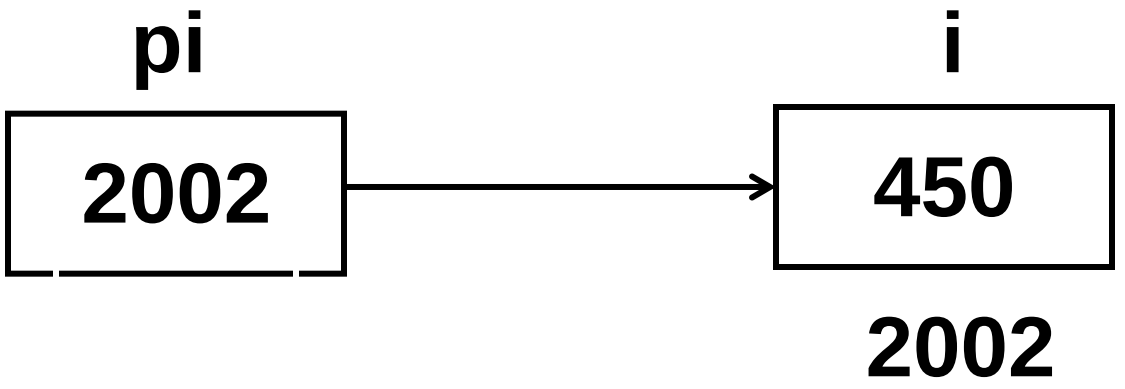
```
int  i = 100 ;
int  * pi ;
```

برنامه‌سازی پیشرفته

pi = & i ;



* pi = 450 ;



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
```

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.

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 ) ;
   printf (“x = %d , y = %d” , x , y) ;
}
```

خروجی:

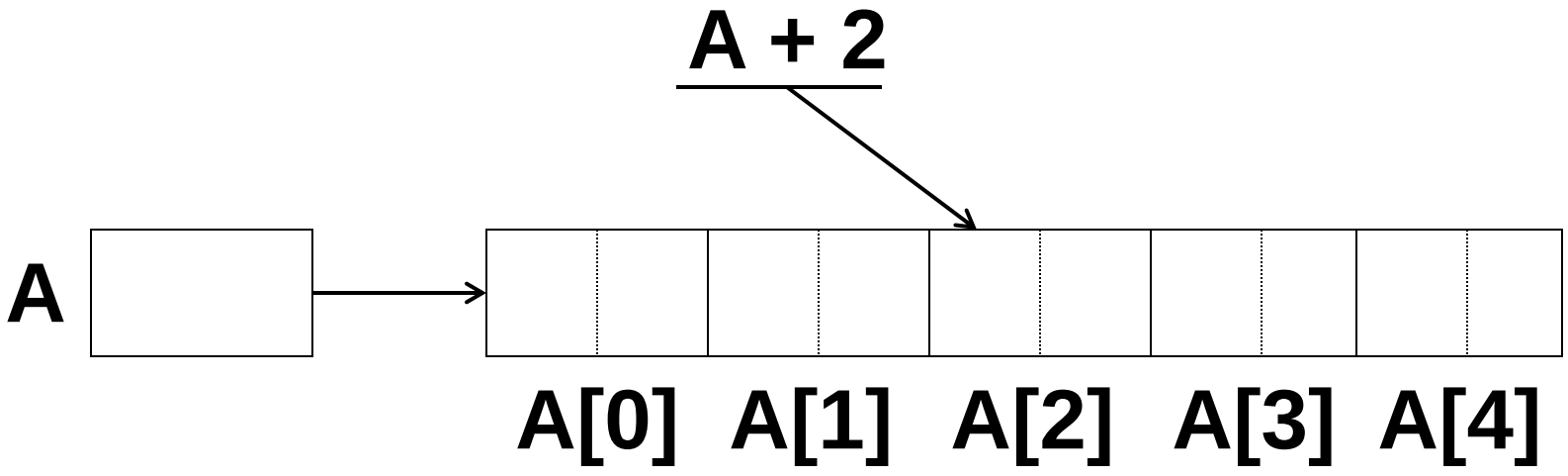
x = 20 , y = 10

برنامه‌سازی پیشرفته

اشاره‌گر و آرایه

نام آرایه اشاره‌گر به اولین عنصر آن است.

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



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) ;
```

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) ) ;
```

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.

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] ;
...
scanf (“%d” , &n) ;
```

برنامه‌سازی پیشرفته

روش صحیح

```
int  n ;
float  *A ;
...
scanf (“%d” , &n) ;
A = (float *) malloc (sizeof (float) * n) ;
... // کار با آرایهٔ A
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 نوع 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.

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”) ;
```

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

مثال: آرایه‌ای برای نگهداری اطلاعات دانشجویان یک کلاس

```
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 ;
```

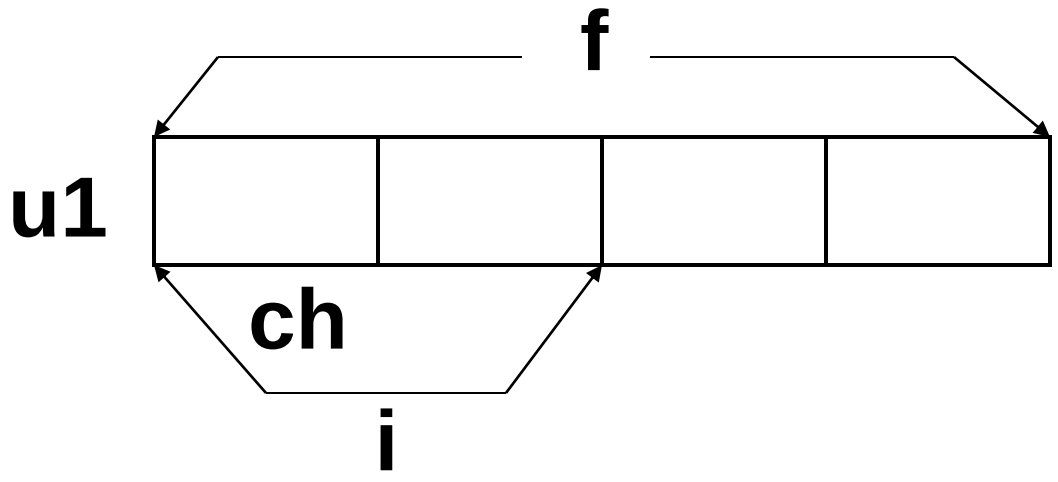
برنامه‌سازی پیشرفته

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.



برنامه‌سازی پیشرفته

جلسهٔ دهم

برنامه‌سازی پیشرفته

فایل

برای انتقال خروجی برنامه‌ها به حافظهٔ پایدار

بدلیل

ماندگاری آنها

ایجاد ارتباط بین برنامه‌ها (فایل بعنوان ورودی)

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.

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>1255346asd66
33<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)

برنامه‌سازی پیشرفته

مثال

◘٩¶•↕☻↓:♣âèÿ‡A€Θ£¥ ↕☻↓:
âèÿ‡A€Θ£¥

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

- 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 انتهای 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

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) ;
```

برنامه‌سازی پیشرفته

جلسهٔ یازدهم

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

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

fread و fwrite

بیشتر برای ورودی – خروجی رکورد استفاده می‌شود

ولی

در همهٔ موارد می‌تواند بکار رود

و

سرعت بالایی نیز دارد.

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 ) ;
```

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.

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) ;
```

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

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

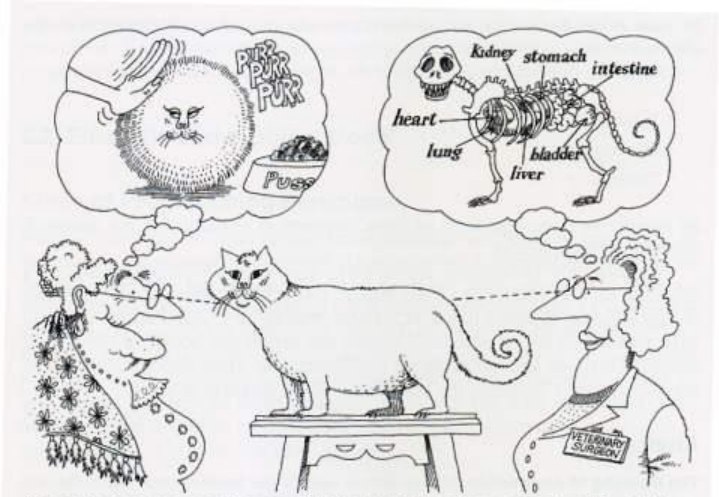
اصول اساسی شی‌گرایی

- تجرید (Abstraction)
- کپسوله‌سازی (Encapsulation)
- ارث‌بری (Inheritance)

برنامه‌سازی پیشرفته

تجرید

استخراج مدلی از مسئله با در نظر گرفتن مهمترین نکات و حذف جزئیات زائد

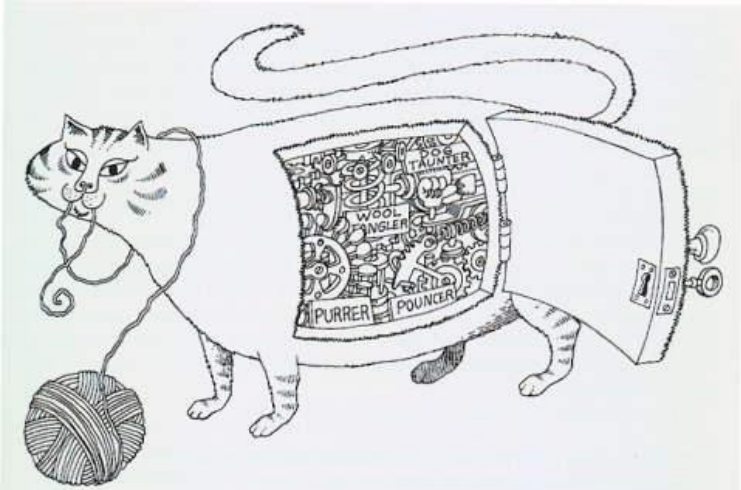


Abstraction focuses upon the essential characteristics of some object, relative to the perspective of the viewer.

برنامه‌سازی پیشرفته

کپسوله‌سازی

مخفی کردن جزئیات پیاده‌سازی از کاربران

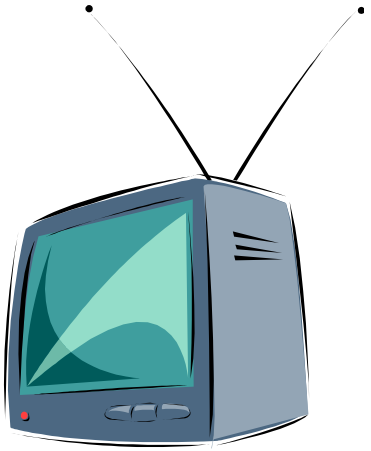


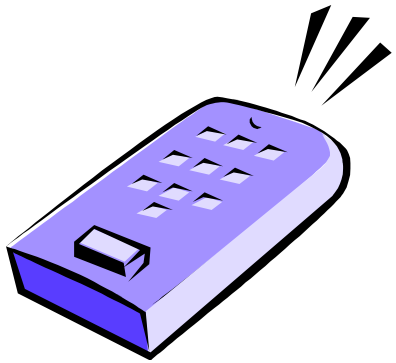
Encapsulation hides the details of the implementation of an object.

برنامه‌سازی پیشرفته

کپسوله‌سازی (ادامه)

کاربران فقط به واسط استفاده وابسته هستند و نه به جزئیات پیاده‌سازی

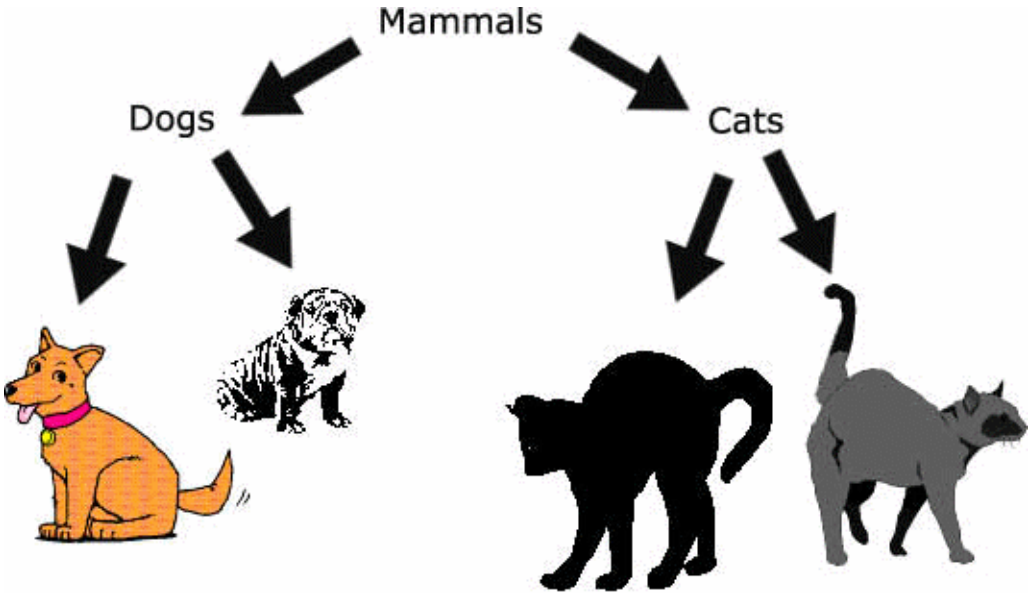




برنامه‌سازی پیشرفته

ارث‌بری

نظم بخشیدن به تجریدهای بدست آمده در ساختاری شبیه به درخت جهت قابلیت استفاده مجدد



برنامه‌سازی پیشرفته

مفاهیم پایه شئ‌گرایی

- شئ (Object)
- کلاس (Class)
- صفت (Property)
- متد (Method)

برنامه‌سازی پیشرفته

Object

موجودیت فیزیکی و یا مفهومی است مانند:

اتوموبیل

چکش

فرآیند شیمیایی

برنامه‌سازی پیشرفته

Object (ادامه)

مجموعه‌ای از داده‌ها + اعمال بر روی آن داده‌ها

مشخصا یک شئ دارای:

- **حالت**
- **رفتار**
- **هویت**

برنامه‌سازی پیشرفته

Object (ادامه)

حالت: وضعیتی که هر لحظه شئ در آن قرار دارد و با مقدار صفات آن در آن لحظه مشخص می‌شود.

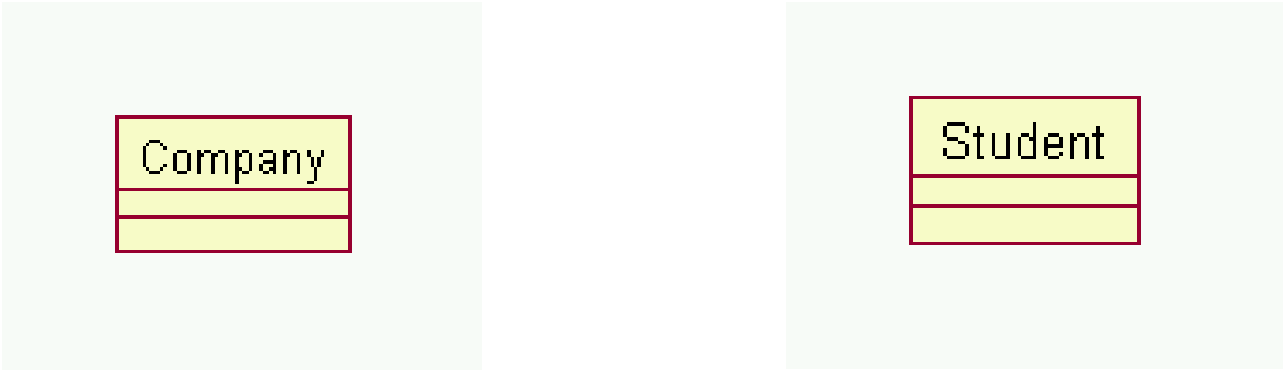
رفتار: روشی که شئ به تعامل با دیگر اشیا می‌پردازد که در واقع همان واسط ارتباطی شئ است.

هویت: همان چیزی است که دو شئ را از یکدیگر متمایز می‌سازد حتی اگر دارای حالت و رفتار یکسان باشند.

برنامه‌سازی پیشرفته

مثالهایی از شئ

اشیا معمولا با مستطیل نمایش داده می‌شوند:



برنامه‌سازی پیشرفته

نمایش اشیاء

با یک مستطیل و اسمی که زیر آن خط کشیده شده نمایش داده می‌شود.

: Lecturer

Y.Welikala

Y.Welikala : Lecturer

برنامه‌سازی پیشرفته

کلاس

مجموعه‌ای از اشیا دارای ویژگی و رفتار یکسان را کلاس می‌نامند.

مانند کلاس:

- **انسان**
- **دانشجو**
- **ماشین**
- **...**

برنامه‌سازی پیشرفته

کلاس (ادامه)

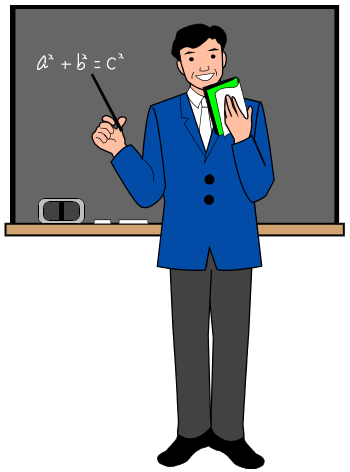
- شی نمونه‌ای (instance) از یک کلاس است.
- کلاس تجریدی است:
  - با تاکید بر ویژگیهای مرتبط
  - نادیده گرفتن ویژگیهای نامرتبط

برنامه‌سازی پیشرفته

کلاس نمونه

•Class

Course



ویژگیها	رفتارها
Name	Add a student
Location	Delete a student
Days offered	Get course roster
Credit hours	Determine if it is full
Start time	
End time	

برنامه‌سازی پیشرفته

نمایش کلاسها

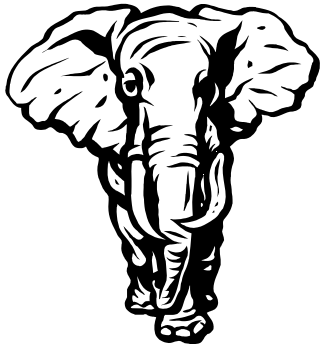
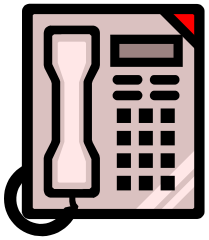
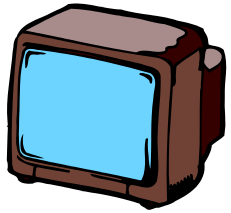
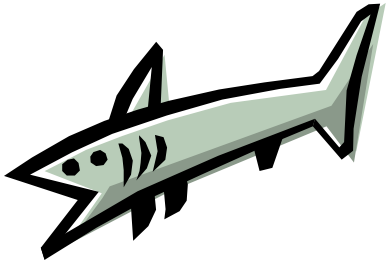
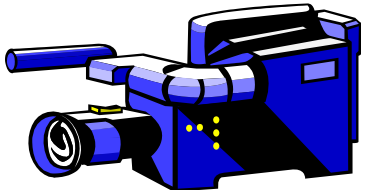
- کلاس با یک مستطیل به صورت زیر نمایش داده مي‌شود.

Lecturer
نام
ذخیره کردن

برنامه‌سازی پیشرفته

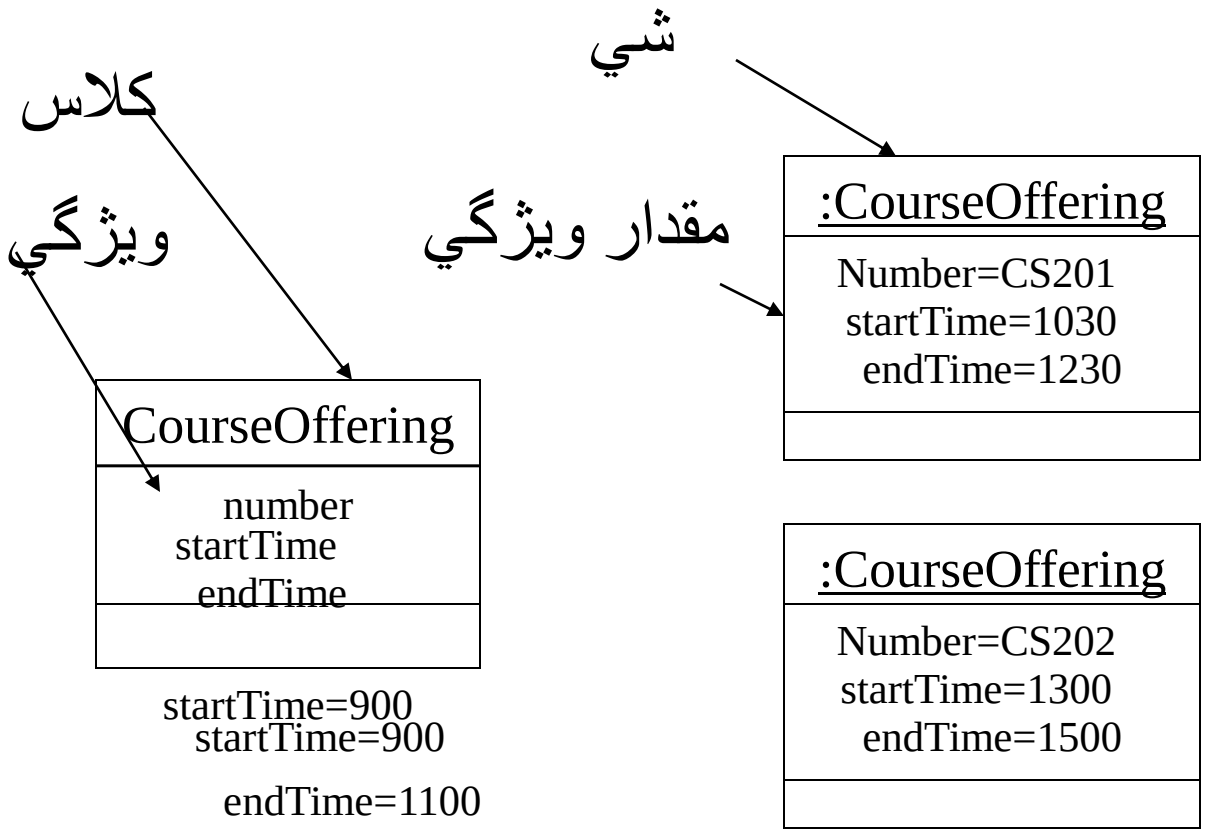
کلاسهایی از اشیا

- چند تا کلاس مشاهده می‌کنید؟

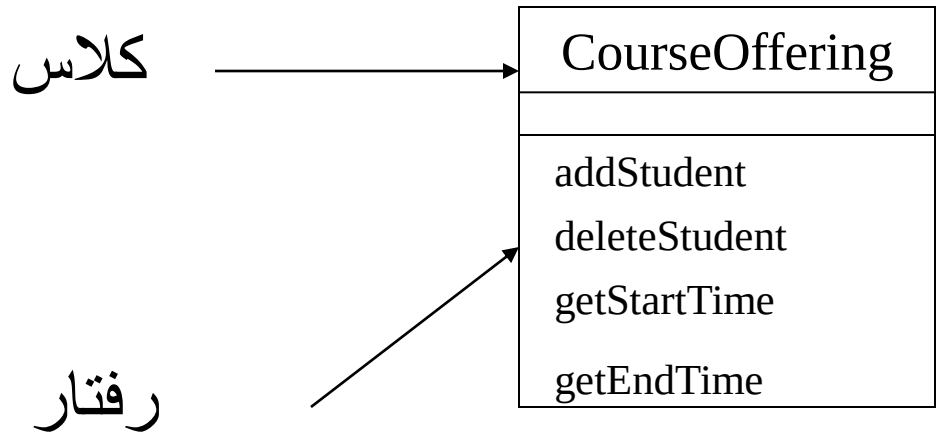
برنامه‌سازی پیشرفته

ویژگي چیست؟



برنامه‌سازی پیشرفته

رفتار چیست؟



برنامه‌سازی پیشرفته

نحوه تعریف کلاس (شکل ساده)

```
class <class_name>          // class header (prototype)
{
public:
   <function_prototypes>
protected:
   <function_prototypes>
private:
   <function_prototypes>
   <data_attributes>
};
```

برنامه‌سازی پیشرفته

مثال

```
Class point
{
    private float x, y;
    public void draw();
}
```

برنامه‌سازی پیشرفته

نحوه تعریف کلاس (شکل ساده)

- کلمات public, private, protected تعیین کنندهٔ نحوه دسترسی استفاده‌کنندگان از کلاس به ویژگیها و رفتار (متدها) کلاس می‌باشد.
- public به معنی دسترسی برای عموم می‌باشد.
- private به معنی دسترسی فقط برای اعضای کلاس می‌باشد.
- protectedبه معنی دسترسی برای اعضای کلاس و ارث‌برندگان کلاس می‌باشد.

برنامه‌سازی پیشرفته

نحوه تعریف کلاس (ارث‌بری)

```
class <class_name> : <superclass_name>
{
public:
   <function_prototypes>
protected:
   <function_prototypes>
private:
   <function_prototypes>
   <data_attributes>
};
```

برنامه‌سازی پیشرفته

سازنده کلاس

در C++ متدی از کلاس که همنام آن کلاس است سازنده کلاس است.

به این معنی که موقع ایجاد شئ از آن کلاس ابتدا متد مذکور اجرا می‌شود.

برنامه‌سازی پیشرفته

مثال

```
Class point
{
    private float x, y;
    public void point(float,float);
    public void draw();
}
```

برنامه‌سازی پیشرفته

مخرب کلاس

در C++ متدی از کلاس که همنام آن کلاس و با یک علامت ~ قبل از آن است مخرب کلاس است.

به این معنی که موقع از بین رفتن شئ آن کلاس در انتها متد مذکور اجرا می‌شود.

برنامه‌سازی پیشرفته

مثال

```
Class point
{
    private float x, y;
    public void ~point();
    public void draw();
}
```

برنامه‌سازی پیشرفته

نحوهٔ ایجاد شئ

با استفاده از اپراتور new

```
Class point {…}
point p = new point(0,10) ;
```

برنامه‌سازی پیشرفته

ورودی در C++

اپراتور ورودی >>

مثال:

```
cin >> var1, var2, var3;
```

برنامه‌سازی پیشرفته

ورودی در C++

<< اپراتور خروجی

مثال:

```
cout << “Text :” << var2 ;
```

برنامه‌سازی پیشرفته

- گیرم که هزار مصحف از بر داری
- با آن چه کنی که نفس کافر داری
- سر را به زمین چه می نهی بهر نماز
- آن را به زمین بنه که در سر داری

برنامه‌سازی پیشرفته

برنامه‌سازی پیشرفته

با سپاس و آرزوی موفقیت

هوشمند عزیزی

مدرس دانشگاه فنی و حرفه ای

۱۳۹۱

برنامه‌سازی پیشرفته

پایان