آموزش نکته به نکته دروس

کنکور کامپیوتر

کاردانی به کارشناسی

- ذخیره و بازیابی اطلاعات
- مدار منطقی
- سیستم عامل
- ساختمان داده‌ها
- زبان تخصصی
- زبان برنامه‌نویسی

گردآوری و تالیف : مهندس علی یگانه ـ مهندس مظاهر مقصودلو

آموزش نکته به نکته دروس

کنکور کامپیوتر

کاردانی به کارشناسی

۱) زبان برنامه‌نویسی C و C++

۲) مدار منطقی

۳) زبان تخصصی

۴) ذخیره و بازیابی اطلاعات

۵) سیستم عامل

۶) ساختمان داده‌ها

کد:۳۰۷۰۸

کارشناسی ناپیوسته

گردآوری و تألیف

مهندس علی یگانه ـ مهندس مظاهر مقصودلو

فهرست مطالب

بخش اول ـ زبان برنامه‌نویسی C و C++

فصل اول: «متغیرها ، ثابت‌ها و انواع داده‌ها»..... ۵
فصل دوم: «عبارات و عملگرها»..... ۶
فصل سوم: «دستورات ورودی و خروجی»..... ۱۳
فصل چهارم: «دستورات شرطی»..... ۱۸
فصل پنجم: «دستورات تکرار»..... ۲۰
فصل ششم: «آرایه‌ها»..... ۲۳
فصل هفتم: «رشته‌ها»..... ۲۵
فصل هشتم: «اشاره‌گرها»..... ۲۷
فصل نهم: «توابع»..... ۳۰
فصل دهم: «ساختار ، اتحاد و کلاسها»..... ۳۶
فصل یازدهم: «فایلها»..... ۴۵

بخش دوم ـ مدار منطقی

فصل اول: «سیستم‌های نمایش اعداد»..... ۵۰
فصل دوم: «جبر بول»..... ۶۴
فصل سوم: «مدارات منطقی ترکیبی»..... ۷۰
فصل چهارم: «مدارات ترکیبی نمونه»..... ۹۱
فصل پنجم: «مدارات ترتیبی»..... ۱۰۶

بخش سوم ـ زبان تخصصی

فصل اول: «Unit 1»..... ۱۲۳
فصل دوم: «Unit 2»..... ۱۲۶
فصل سوم: «Unit 3»..... ۱۳۰
فصل چهارم: «Unit 4»..... ۱۳۴
فصل پنجم: «Unit 5 »..... ۱۳۶
فصل ششم: «Unit 6 »..... ۱۳۹
فصل هفتم: «Unit 7 »..... ۱۴۲
فصل هشتم: «Unit 8 »..... ۱۴۷
فصل نهم: «Unit 9 »..... ۱۵۰
فصل دهم: «Unit 10 »..... ۱۵۴

بخش چهارم ـ ذخیره و بازیابی اطلاعات

فصل اول: «حافظه‌ها و دستگاه‌های ذخیره و بازیابی»..... ۱۵۹
فصل دوم: «مفاهیم پایه‌ایی»..... ۱۶۹
فصل سوم: «ارزیابی پارامترهای رسانه‌ها» ۱۸۸
فصل چهارم: «سیستم و ساختار فایل»..... ۱۹۹
فصل پنجم: « شاخص»..... ۲۱۴

بخش پنجم ـ سیستم عامل

فصل اول: «تعریف سیستم عامل و انواع وقفه‌ها»..... ۲۳۸
فصل دوم: «دیاگرام پردازش»..... ۲۴۵
فصل سوم: «الگوریتمهای زمانبندی»..... ۲۴۷
فصل چهارم: «مدیریت حافظه و حافظه مجازی» ۲۶۱
فصل پنجم: «بن بست»..... ۲۷۵
فصل ششم: «سیستم عامل لینوکس..... ۲۸۰

بخش ششم ـ ساختمان داده‌ها

فصل اول: «تجزیه و تحلیل الگوریتمها»..... ۲۸۵
فصل دوم: «توابع بازگشتی»..... ۲۸۸
فصل سوم: «ماتریسها»..... ۲۹۰
فصل چهارم: «پشته»..... ۲۹۵
فصل پنجم: «صف»..... ۳۰۳
فصل ششم: «لیستهای پیوندی»..... ۳۰۵
فصل هفتم: «درختها»..... ۳۱۲
فصل هشتم: «درختهای ویژه»..... ۳۲۱
فصل نهم: «گرافها»..... ۳۲۵
فصل دهم: «مرتب‌سازی»..... ۳۳۱

بخش اول

زبان برنامه‌نویسی C و C++

فصل اول

«متغیرها . ثابتها و انواع داده‌ها»

شناسه‌ها و کلمات کلیدی

منظور از شناسه‌ها (identifiers) نام‌هایی است که به توابع، متغیرها و غیره داده می‌شود. نام شناسه در توربو C می‌تواند یک تا ۳۱ کاراکتر و نام یک متغیر یا شناسه در C++ می‌تواند تا ۱۰۲۴ کاراکتر باشد. قوانین نام‌گذاری شناسه‌ها در زبان C و C++ عبارتست از :

۱- اولین کاراکتر شناسه‌ها باید یک حرف یا زیرخط باشد.
۲- از کاراکتر دوم به بعد می‌توان از حروف، اعداد یا زیرخط استفاده نمود.
۳- در نام شناسه‌ها نمی‌توان از blank و کاراکترهای غیرمجاز مثل ؟ ، ! ، $ و ... استفاده کرد.
۴- در نام شناسه نمی‌توان از کلمات کلیدی (keywords) زبان C و C++ استفاده نمود.

تذکر : نام دو متغیر (یا یک متغیر و یک زیر برنامه) در داخل یک بلوک نمی‌تواند یکسان باشد ولی در دو بلوک مختلف می‌توان از یک نام، برای دو متغیر (یا یک متغیر و یک زیر برنامه) استفاده کرد.

انواع داده‌ها در زبان C

داده‌های زبان C به ۵ دسته اصلی زیر تقسیم می‌شوند:

نوع داده	جهت ذخیره	مقدار فضای مورد نیاز	محدوده عددی
char	کاراکتر یا عدد صحیح	۱ بایت	-128 تا +127
int	عدد صحیح	۲ بایت	-32768 تا +32767
float	اعشاری	۴ بایت	3.4E-38 تا 3.4E+38
double	اعشاری با دقت مضاعف	۸ بایت	1.7E - 308 تا 1.7E+308
void	---	---	---

تذکر : در زبان C++ دو نوع داده جدید به نام‌های bool و wchar_t اضافه شده‌اند. نوع bool مانند boolean زبان پاسکال می‌تواند مقادیر true و false را بپذیرد و نوع wchar_t جهت معرفی کاراکترهای ۱۶ بیتی به کار می رود.

نکته : unsigned , signed *فقط با* int *و* char *به کار برده می‌شوند، اگر آن‌ها را پشت* float *یا* double *بنویسید کامپایلر پیغام خطا می‌دهد.*

نکته : short *و* long *می‌توانند پشت هر ٤ نوع* (double , float , char , int) *بیایند. به صورت پیش‌فرض متغیری که از نوع* int *یا* char *تعریف می‌شود، از نوع علامت‌دار است* (signed) *یعنی می‌تواند عدد مثبت یا منفی را در خود ذخیره کند. اگر قبل از* int *یا* char unsigned *عبارت آورده شود آنگاه آن متغیر فقط می‌تواند عدد مثبت را ذخیره کند و البته محدوده آن افزایش می‌یابد.*

نکته : unsigned k ; *معادل* unsigned int k ; *می‌باشد.*

نکته : long k ; *معادل* long int k ; *می‌باشد.*

مقادیر ثابت (constants)

مقادیر ثابت به ۴ دسته اصلی زیر تقسیم می‌شوند:

۱ـ مقادیر صحیح : مقادیر صحیح در سه مبنای ۱۰ (Decimal)، مبنای ۸ (octal) و مبنای ۱۶ (hexadecimal) نمایش داده می‌شوند. در داخل اعداد صحیح نباید نقطه به کار برد، همچنین اعداد دسیمال نباید با صفر شروع شوند. قبل از اعداد در مبنای ۱۶، از علامت 0x **یا** 0X استفاده می‌گردد (مانند : 0x7Fab و 0X5 و 0x1). قبل از اعداد در مبنای ۸ ، از یک عدد صفر استفاده می‌شود (مانند : 017 و 042).

۲ـ مقادیر اعشاری : اعداد ثابت اعشاری به وسیله نقطه (.) یا به صورت نماد علمی (با حروف E یا e) نمایش داده می‌شوند، مانند :

1.0 1. 0.2 827.625
2E-8 0.07e-3 1.2e2 1.2e+2

تذکر : عدد بعد از e در نماد علمی حتماً باید صحیح باشد، در صورتی که اعشاری باشد پیام خطا صادر می‌شود.

۳ـ مقادیر کاراکتری : کاراکترهای ثابت داخل علامت تک گیومه ' ' نوشته می‌شوند، مانند : 'a' ، '?' .

***نکته :** ثابت‌های کاراکتری حداکثر می‌توانند ۲ کاراکتر بپذیرند، ولی فقط کاراکتر اول آن‌ها در نظر گرفته می‌شود.*

۴ـ مقادیر رشته‌ای : مقادیر ثابت رشته‌ای داخل علامت " " نوشته می‌شوند، مثل : "book" .

متغیرها

مکانی از حافظه که برای نگهداری موقتی داده‌ها استفاده می‌شود، اصطلاحاً متغیر گفته می‌شود که در زبان C و C++ به صورت زیر تعریف و اعلان می‌شوند :

```
نوع داده      نام متغیر ;
int           x ;
```

تذکر : در صورتی که در اعلان متغیرها، تعدادی از متغیرها از یک نوع باشند می‌توان آن‌ها را با علامت کاما (,) و در یک اعلان ذکر نمود، مانند :

```
flaot     x , y , z ;
```

تذکر : در زبان C و C++ ، هم‌زمان با تعریف متغیرها می‌توان آن‌ها را مقداردهی نیز نمود، مانند :

```
int     a = 20 ;
```

اعلان ثابت‌ها

ثابت‌ها، مقادیری هستند که مقدار آن‌ها در طول اجرای برنامه تغییر نمی‌یابد و در زبان C و C++ ، ثابت‌ها به دو صورت زیر تعریف می‌شوند :

۱- با استفاده از کلمه کلیدی const ۲- با استفاده از ماکرو # define

اعلان ثابت‌ها با استفاده از کلمه کلیدی const

فرم کلی تعریف ثابت‌ها با استفاده از کلمه کلیدی const به صورت زیر است :

```
const [ نوع داده ] نام ثابت = مقدار ;
const    float    a  =  17.25 ;
```

تذکر : اگر در تعریف ثابت‌ها، نوع داده ثابت ذکر نشود، آنگاه ثابت به طور پیش‌فرض از نوع داده int در نظر گرفته می‌شود.

اعلان ثابت‌ها با استفاده از ماکرو # define

فرم کلی تعریف ثابت‌ها با ماکرو # define به صورت زیر است :

```
# define    نام ثابت    مقدار
# define      a      17.25
```

تذکر : اگر ثابتی با استفاده از ماکرو # define تعریف شود، آنگاه هیچ حافظه‌ای اشغال نخواهد کرد.

دستور typedef

با این دستور می‌توان یک نام مجازی به یک نوع داده‌ای داد و یا نام جدید برای یک نوع از پیش تعریف شده ایجاد کرد. فرم کلی این دستور به صورت زیر است :

```
typedef    نوع داده    نام جدید ;
typedef  int  integer ;
integer  n ;
```

n متغیری از نوع داده صحیح است.

فصل دوم

«عبارات و عملگرها»

انواع عملگرهای زبان C و C++ عبارتست از :

۱- عملگر جایگزینی (assignment)
۲- عملگر باینری (binary)
۳- عملگر یکتایی (unary)
۴- عملگر منطقی (logical)
۵- عملگر رابطه‌ای (relational)
۶- عملگر بیتی (bitwise operators)
۷- عملگر شرطی (؟)
۸- عملگر کاما (,)

عملگر جایگزینی

علامت جایگزین (assignment) در زبان C ، مساوی (=) می‌باشد که عبارت سمت راست را در متغیر سمت چپ می‌ریزد. سمت چپ فقط می‌تواند نام یک متغیر باشد.

در زبان C نوع عبارت سمت راست می‌تواند با نوع متغیر سمت چپ متفاوت باشد. در این حال مقدار عبارت سمت راست تبدیل به نوع متغیر سمت چپ می‌شود.

نکته : اگر یک مقدار اعشاری به متغیر صحیح نسبت داده شود قسمت اعشار آن حذف می شود.

نکته :هنگامی که عبارت صحیح ۲ بایتی یا ٤ بایتی به یک متغیر کاراکتری یک بایتی نسبت داده شود، تنها یک بایت سمت راست عدد صحیح در متغیر کاراکتری ریخته می‌شود و بایت‌های دیگر سمت چپ عبارت از بین می‌رود.

نکته :هنگامی که عبارت double در متغیر float ای ریخته شود، دقت کاهش می‌یابد، یعنی تعدادی از اعشار حذف شده و حاصل گرد می‌شود.

مثال:

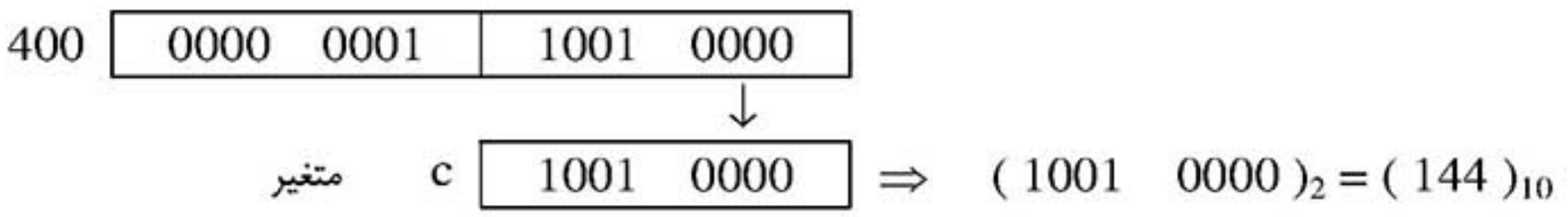
```
char c = ' a ' ;
int i = 52 ;
float f ;
f = i ;          f برابر 52.0 می‌شود.
i = c ;          i برابر 65 می‌شود.
f = c ;          f برابر 65.0 می‌شود.
```

مثال: *خروجی برنامه زیر چه می‌شود ؟*

```
main ( )
{ unsigned char c ;
c = 100 * 4 ;
printf ( " % d " , c ) ; }
```

خروجی عدد ۱۴۴ می‌شود و علت آن به صورت زیر است :

4 * 100 برابر 400 می‌شود که حاصل در یک بایت جا نمی‌گیرد و حداقل ۲ بایت فضا نیاز دارد.



عملگر دوتایی باینری (Binary) / ریاضی

عملگرهای ریاضی عبارتند از :

عملکرد	عملگر
جمع	+
تفریق	−
ضرب	*
تقسیم	/
باقی‌مانده صحیح	%

نکته: اپراتور % باقی‌مانده تقسیم صحیح دو عملوند خود را برمی‌گرداند. دو عملوند اپراتور % باید حتماً مقادیر یا متغیرهایی صحیح باشند (مثل long , char , int)، در غیر این صورت خطای کامپایلری رخ می‌دهد، همچنین عملوند دوم نمی‌تواند صفر باشد. اپراتور % در زبان C مشابه اپراتور mod در زبان پاسکال است.

```
printf ( " % d " , 15 % 6 ) ; → 3
```

نکته : اپراتور / هم تقسیم اعشاری و هم تقسیم صحیح را انجام می‌دهد. اگر هر دو عملوند / عباراتی صحیح باشند، آنگاه / تقسیم صحیح را انجام می‌دهد، ولی اگر حداقل یکی از عملوندهای / اعشاری باشند، / تقسیم اعشاری را انجام می‌دهد.

مثال:

```
1)  int  a = 20 , b = 3 ;
    printf ( " % d " , a / b ) ; →  6
2)  float a = 20 , b = 3 ;
    printf ( " % f " , a / b ) ; →  6.666667
```

تذکر : اگر در یک عبارت، چند عملگر دارای اولویت یکسانی باشد. آنگاه اولویت به عملگر سمت چپ داده خواهد شد.

عملگر یکتایی (Unary)

این اپراتورها که فقط یک عملوند دارند، عبارتند از : – ، – – ، + + . عملوند متعلق به این عملگرها می‌تواند، صحیح، کاراکتری یا اعشاری باشد. عملگر + + (افزایش increment) یک واحد به عملوند خود اضافه می‌کند و عملگر – – (کاهش decrement) یک واحد از عملوند خود کم می‌کند. به عبارتی دیگر :

```
+ + i ;    معادل   i = i + 1 ;
- - i  ;    معادل   i = i – 1 ;
```

مثال: ***با اجرای تکه برنامه زیر مقدار متغیرهای x و y چند است؟***

```
1) x = 10 ;
   y = + + x ;
```

x برابر ۱۱ و y نیز برابر ۱۱ می‌شود. اول یک واحد به x اضافه می‌شود، سپس x در y ریخته می‌شود.

```
2) x = 10 ;
   y = x + + ;
```

x برابر ۱۱ و y برابر ۱۰ می‌شود. اول x در y ریخته می‌شود، سپس یک واحد به x اضافه می‌شود.

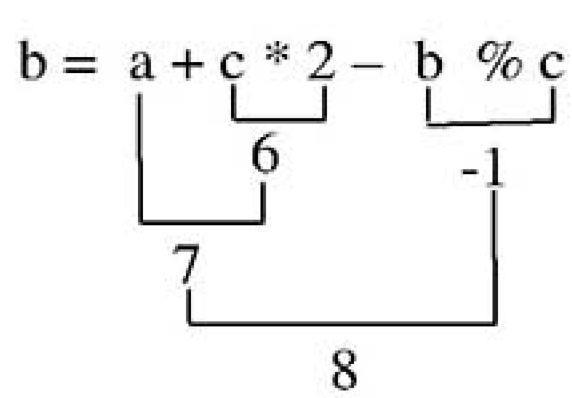
نکته : اگر در عبارتی -- و یا ++ وجود داشت، آنگاه باید عبارت را براساس قواعد زیر محاسبه کنیم :

۱- ابتدا تمام -- و ++ سمت چپ را محاسبه کنیم.
۲- عبارت را بدون -- و ++ سمت چپ و سمت راست محاسبه کنیم.
۳- در نهایت -- و ++ سمت راست را محاسبه کنیم.

مثال: ***با اجرای تکه برنامه زیر مقدار متغیرهای b , a و c چند می‌شود ؟***

```
a = 2  ;  b = -1  ;  c = 3  ;
b = - - a + c * 2 – b + + % c ++;
```

ابتدا باید a - - را اعمال کنیم که در این حالت مقدار جدید متغیر a معادل ۱ خواهد شد. حال باید کل عبارت را بدون -- و ++ سمت چپ و راست حل کنیم، که داریم :



با محاسبه عبارت مقدار جدید متغیر b معادل ۸ خواهد شد. حال باید -- و ++ سمت راست را اعمال کنیم که با این عمل مقدار متغیر b معادل ۹ و مقدار متغیر c معادل ۴ خواهد شد.

عملگرهای منطقی

این عملگرها در عبارات منطقی استفاده می‌شوند که عبارتند از :

عملکرد	عملگر
عملگر AND منطقی	& &
عملگر OR منطقی	| |
عملگر NOT منطقی	!

حاصل این عملگر ها یا صفر (ارزش نادرستی) است و یا یک (ارزش درستی) .

مثال: *حاصل عبارت زیر چیست؟*

1) $5 \&\& 0 = 0$
2) $2 \mid\mid -2 = 1$
3) $!10 = 0$

عملگرهای رابطه‌ای

برای مقایسات و شروط از عملگرهای رابطه‌ای استفاده می‌شود که عبارتند از :

عملکرد	عملگر
بزرگ‌تر	>
کوچک‌تر	<
بزرگ‌تر مساوی	>=
کوچک‌تر مساوی	<=
مساوی	==
نامساوی (مخالف)	!=

حاصل این عملگرها یکی از دو مقدار درستی (عدد غیرصفر) و یا نادرستی (عدد صفر) می‌باشد.

مثال:

$3 * 4 != 5 * 2$

12 10 (عدد غیرصفر معادل ارزش درستی است)

1

عملگر بیتی (Bitwise Operators)

عملگرهای بیتی فقط به همراه نوع داده‌های صحیح استفاده می‌شوند و نمی‌توان آن‌ها را با نوع داده‌های دیگری مثل اعشاری، رشته‌ای و یا اشاره گرها به کار برد. عملگرهای بیتی عبارتند از :

عملکرد	عملگرهای بیتی در زبان C
AND بیتی	&
OR بیتی	|
XOR بیتی	^
NOT بیتی	~
شیفت به راست	>>
شیفت به چپ	<<

هر گاه علمگرهای بیتی در کنار اعداد بیایند، ابتدا می‌بایست اعداد را تبدیل به مبنای ۲ کرده و سپس بیت به بیت آن‌ها را با یکدیگر AND , OR , XOR و ... کنیم.

مثال: *خروجی تکه برنامه زیر چیست ؟*

```
unsigned char m ;
m = 129 | 3 ;
printf ( " % d " , m ) ;
```

```
129   →   1000  0001
3     →   0000  0011
          ----------
          1000  0011   →  131
```

پس خروجی برنامه عدد 131 می‌شود.

نکته : *اگر در یک عبارت داشته باشیم* $\sim x$ *، آنگاه حاصل از رابطه مقابل محاسبه می‌شود:* $\sim x = -(x+1)$

نکته : *اگر در یک عبارت داشته باشیم* $a >> b$*، آنگاه حاصل از رابطه* $\left[\frac{a}{2^b}\right]$ *محاسبه می‌شود.*

نکته : *اگر در یک عبارت داشته باشیم* $a << b$ *آنگاه حاصل از رابطه* $a * 2^b$ *محاسبه می‌شود.*

نکته : *XOR هر عدد با خودش صفر می‌شود.*

مثال: *خروجی تکه برنامه‌های زیر چیست؟*

1) `printf ( " % D " , 53 >> 3 ) ;` $\xrightarrow{\text{خروجی}}$ 6
2) `printf ( " % D " , 53 ^ 53 ) ;` $\xrightarrow{\text{خروجی}}$ 0

نکته : ***اگر مقدار حاصل از یک عملیات دوباره با همان مقدار قبلی،*** XOR ***شود، آنگاه مقدار اولیه به دست می‌آید.***

مثال: ***خروجی برنامه زیر چیست ؟***

```
int  x = 7 ;
printf ( " % d " , x&&8 ) ; → 1
printf ( " % d " , x&8 ) ; → 0
```

در x&&8 عدد 8 درست و 7 درست تعبیر می‌شود و نتیجه درست یعنی 1 می‌شود. در دستور چاپ دوم داریم :

```
7  →   0000 0111   AND
8  →   0000 1000
       ---------  حاصل →  0  عدد
       0000 0000
```

عملگر شرطی (: ?)

فرم کلی استفاده از این عملگر به صورت زیر است :

عبارت ۲ : عبارت ۱ ? شرط

ابتدا شرط بررسی می‌شود، اگر شرط درست باشد (یعنی مقداری غیر صفر داشته باشد) عبارت ۱ محاسبه شده و نتیجه آن برگردانده می‌شود، در غیر این صورت نتیجه عبارت ۲ برگردانده می‌شود.

مثال: ***حاصل عبارت زیر چیست؟***

```
flag = ( I < 0 )  ?  0 : 100 ;
```

اگر ۱ عدد منفی باشد، مقدار flag صفر شده، در غیر این صورت مقدار flag برابر ۱۰۰ می‌شود.

خلاصه‌نویسی در عبارات جایگزینی

در زبان C می‌توان چند عبارت = را در یک جمله نوشت :

مثال:

```
int  i , j ;
i = j = 5 ;
```

ابتدا ۵ داخل j و سپس j داخل i ریخته می‌شود، توجه کنید عملیات از راست به چپ انجام می‌گیرد. نوع متغیر در طرفین = می‌تواند مختلف باشد که در این حالت مشابه قوانین تبدیل نوع که قبلاً بیان شد، عمل می‌شود.

زبان C دارای یک طرز نوشتن خلاصه برای جملات جایگزینی می‌باشد، بدین معنا که مثلاً دستور x = x + 10 ; می‌تواند به صورت x + = 10 ; نیز نوشته شود، این طرز نوشتن برای تمامی عملگرهای دوتایی قابل استفاده است. پس عملگرهای += ، -= ، *= ، /= ، %= ، &= ، ^= ، |= ، >>= ، <<= همگی درست هستند، مانند:

$i += 5 ; \xrightarrow{\text{معادل}} i = i + 5 ;$

$f -= g ; \xrightarrow{\text{معادل}} f = f - g ;$

$j *= (I - 3) ; \xrightarrow{\text{معادل}} j = j * (i - 3) ;$

$j *= i - 3 ; \xrightarrow{\text{معادل}} j = j * (i - 3) ;$

عملگر کاما (,)

عملگر کاما (comma operator) باعث می‌شود ترتیبی از عملیات انجام شود. اگر این علامت در سمت راست یک عبارت جایگزینی استفاده گردد، حاصل آخرین عبارت سمت راست در متغیر سمت چپ کپی می‌شود.

مثال: ***حاصل عبارت زیر چیست ؟***

```
a = 10 ;
b = ( a = a - 5 , 30 / a ) ;
```

پس از اجرای دو دستور فوق a برابر ۵ و b برابر ۶ می‌شود.