درس «مبانی کامپیوتر و برنامه‌سازی»

متغیر، نوع داده و عملگر

صادق علی‌اکبری

سرفصل مطالب

- مفهوم متغیر
- انواع داده
- عملگرها
- توابع ورودی و خروجی
  - scanf
  - printf

داده‌های در برنامه‌ها

- برنامه‌نویسی: یک روش حل مسأله
- داده: بخشی از اطلاعات که در فرایند حل مسأله از آن استفاده می‌کنیم
- یک برنامه کامپیوتری با انواع مختلفی از داده‌ها سروکار دارد
  - عدد، کاراکتر، متن و ...
  - مانند: نام، سن و معدل دانشجو

مفهوم متغیر

- یک برنامه کامپیوتری، برای کار با هر داده، آن را با یک نام مشخص می‌کند
- متغیر (variable): اسمی نمادین که یک داده را در برنامه مشخص می‌کند
  - با کمک نام متغیر به آن داده مراجعه می‌کنیم
  - چرا عنوان «متغیر» استفاده می‌شود؟
    - چون مقدار این داده قابل تغییر است
- مثال: متغیر age برای نگهداری سن دانشجو
- یادآوری: مفهوم متغیر در ریاضیات

مثال برای متغیرها

```
int main()
{
   int year ;
   int birth ;
   int age ;

   year = 2015;
   birth = 1983;

   age = year - birth;

   return 0;
}
```

- در این برنامه سه متغیر وجود دارد:
- year, birth, age
- نوع این متغیرها؟
- عدد صحیح (int)

متغیر (Variable)

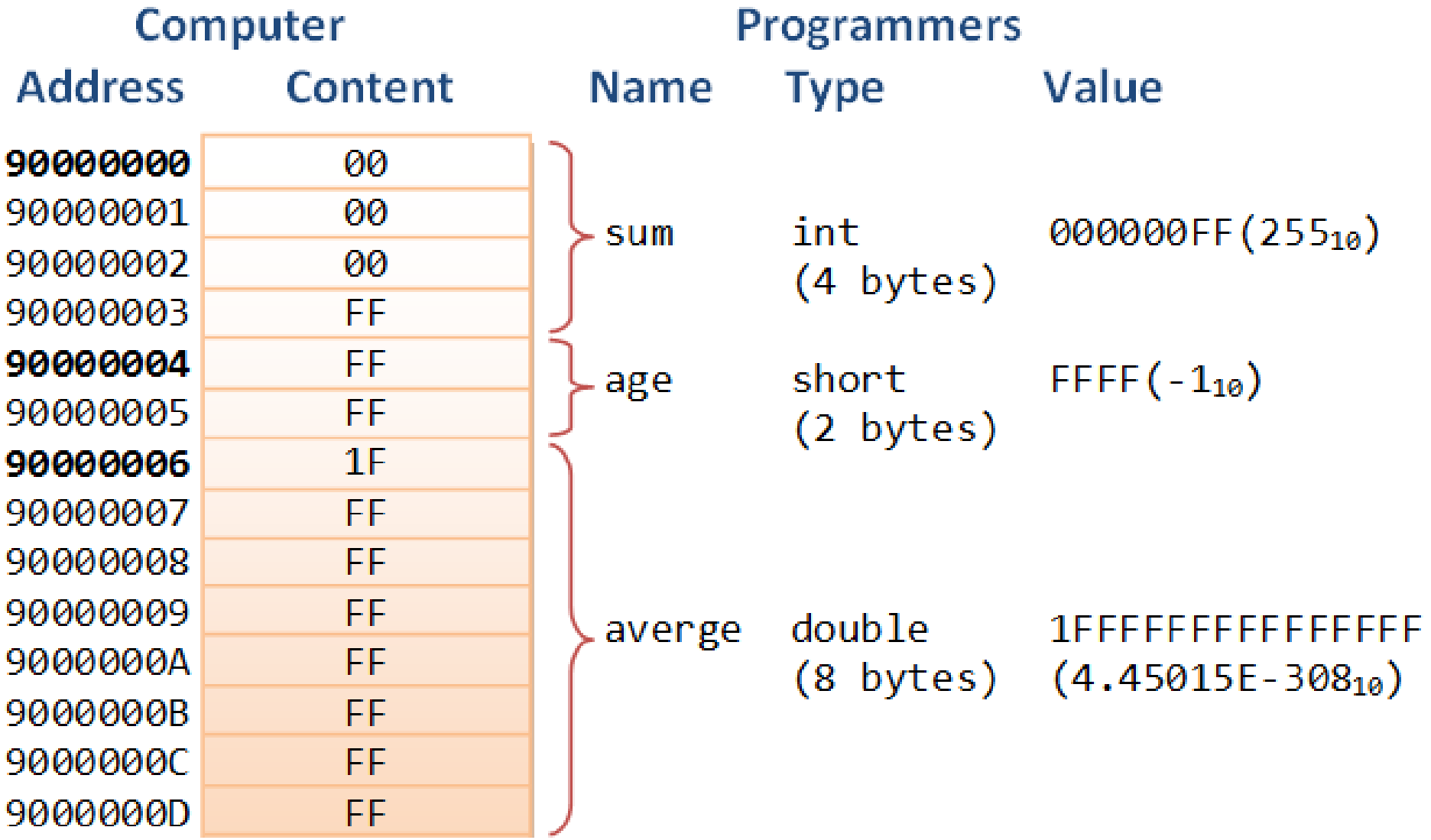
- هر متغیر **نام** دارد (name)
  - مثلاً متغیری با نام age برای نگهداری سن دانشجو
- هر متغیر یک **نوع** دارد (data type)
  - مثلاً نوع متغیر age ، عدد صحیح (int) است
  - انواع داده: عدد صحیح، عدد اعشاری، کاراکتر، متن و ...
- هر متغیر یک **مقدار** دارد (value)
  - در زمان اجرای برنامه، در هر لحظه، هر متغیر مقداری دارد
  - مثلاً مقدار متغیر age در انتهای برنامه قبلی، ۳۲ بود
- هر متغیر، برای نگهداری از مقدار خود، (معمولاً) بخشی از **حافظه** را اشغال می‌کند

مقدار متغیرها

- مقدار متغیرها کجا نگهداری می‌شود؟
- هر برنامه کامپیوتری، برای کار با داده‌ها، آن‌ها را در حافظه نگه می‌دارد
  - بخشی از حافظه برای هر داده اختصاص می‌یابد
  - هر متغیر در بخشی از حافظه نگهداری می‌شود

    (بخشی از حافظه را اشغال می‌کند)
  - آدرس متغیر: آدرس (ابتدای) محلی از حافظه که مسؤول نگهداری از مقدار آن است

متغیرها در حافظه



مثال

```
int main()
{
  int fib1, fib2, fib3 ;
  fib1 = 1;
  fib2=1;
  fib3=fib1+ fib2;

  fib1=fib2;
  fib2=fib3;
  fib3=fib1+ fib2;

  fib1=fib2;
  fib2=fib3;
  fib3=fib1+ fib2;

  fib1=fib2;
  fib2=fib3;
  fib3=fib1+ fib2;
  return 0;
}
```

نامگذاری متغیرها: بایدها

- عنوان (نام یا شناسه) متغیرها باید از قاعده خاصی تبعیت کند
- مثلاً age یک نام معتبر و 1a یا $2 نام‌هایی نامعتبر برای متغیرها هستند
- قوانین نامگذاری:

۱- نام متغیر باید از حروف بزرگ یا کوچک انگلیسی، ارقام یا آندرلاین (_) تشکیل شود

۲- نام متغیر با رقم شروع نشود

۳- نام متغیر یک کلمه رزرو شده در زبان C نباشد (Reserved Word)

- نکته: برخی کلمات در هر زبان، برای کاربردهای خاصی رزرو شده‌اند
  - مانند int, if, for, double, float, ...

مثال: کدام نام‌ها صحیح هستند؟

- نام‌های غلط:
  - number 1
  - num 1
  - addition of program
  - 1num
  - 1_num
  - 365_days
  - int
  - char
  - continue
- نام‌های صحیح:
  - num
  - Num
  - Num1
  - _NUM
  - NUM_temp2
  - age
  - personAge
  - maxGradePossible

نامگذاری مناسب متغیرها

- نام‌های گویا و شفافی برای متغیرها انتخاب کنید
- از اسامی مبهم و کوتاه پرهیز کنید
- مثلاً این‌ها اسمهای خوبی نیستند:
- a, x, number, a1, a2, …
- این‌ها اسم‌های بهتری هستند:
- age, personName, maxGrade, …
- چرا؟
- برنامه‌ای خوب است که راحت خوانده شود و به سادگی فهمیده شود
  - چرا؟

نکته: زبان C حساس به حروف کوچک و بزرگ است

```
int main()
{
 int number;
 int NUMBER;
 number = 2;
 NUMBER = 3;
 return 0;
}
```

- زبان C (و C++) اصطلاحاً Case Sensitive هستند
- مثلاً age و AGE دو نام متفاوت هستند
- اشکال برنامه زیر چیست؟

```
#include <stdio.h>
int main()
{
 PRINTF("Salam");
 return 0;
}
```

نکته

- در اثر کامپایل یک برنامه، نام متغیرها جای خود را به آدرس متغیرها می‌دهند
- به عبارت دیگر: در برنامه اجرایی اثری از نام متغیرها نیست
- مثلاً در فایل a.exe که از کامپایل a.cpp حاصل می‌شود:
  - نام متغیرها در برنامه اجرایی (مثلاً فایل exe) ذخیره نمی‌شود

ویژگی‌های متغیرها

- هر متغیر دارای این ویژگی‌ها است:
  - نوع داده (data type)
  - نام (name)
  - آدرس حافظه (address)
  - مقدار (value)
- نام و نوع متغیرها در زمان برنامه‌نویسی مشخص می‌شود: Compile-time
  - نام و نوع متغیر قابل تغییر نیست
- آدرس و مقدار متغیرها در زمان اجرا مشخص می‌شود: Runtime

تمرین

- برنامه‌ای بنویسید که
  - طول و عرض یک مستطیل را در دو متغیر نگه دارد
  - و محیط مستطیل را محاسبه کند و در متغیری دیگر ذخیره سازد
  - مثلاً فرض کنید طول و عرض مستطیل ۱۰ و ۷ است

```
int main()
{
 int length, width, perimeter;
 length = 10;
 width = 7;
 perimeter = length + width + length + width ;
 return 0;
}
```

- اسامی مناسبی برای متغیرها انتخاب کرده بودید؟

انواع داده (Data Types)

نوع داده

- یادآوری: هر متغیر نام، نوع، آدرس و مقدار دارد
- نوع داده (data type)
- زبان C انواع مختلفی از داده‌ها را پشتیبانی می‌کند
- برخی از مهمترین انواع داده در زبان C: int, char, float
- مثال:

```
int age = 32;
char sharp = '#';
float pi = 3.14;
```

انواع داده در زبان C++

- نوع داده عدد صحیح: int
- انواع داده عدد اعشاری: float و double
  - نوع double بازه گسترده‌تری را پوشش می‌دهد
- انواع داده کاراکتر: char و wchar_t
  - نوع داده wchar_t برای کاراکترهای فراتر از اَسکی (مثل یونیکد)
- نوع داده بولین (منطقی) : bool
  - به صورت عادی در انواع داده‌های زبان C نیست
- نوع داده بی‌مقدار: void

مشخصات انواع داده

- برای برخی از انواع داده، با یکی از کلیدواژه‌های زیر می‌توانیم نوع داده را دقیق‌تر مشخص کنیم:

۱- signed یا unsigned

- مشخص می‌کنند که عدد موردنظر می‌تواند علامت‌دار باشد یا خیر

۲- short یا long

- بازه عدد موردنظر را مشخص می‌کنند

(تعداد بایت حافظه مورد نیاز)

Type	Typical Bit Width	Typical Range
char	1byte	-127 to 127 or 0 to 255
unsigned char	1byte	0 to 255
signed char	1byte	-127 to 127
int	4bytes	-2147483648 to 2147483647
unsigned int	4bytes	0 to 4294967295
signed int	4bytes	-2147483648 to 2147483647
short int	2bytes	-32768 to 32767
unsigned short int	2bytes	0 to 65,535
signed short int	2bytes	-32768 to 32767
long int	4bytes	-2,147,483,648 to 2,147,483,647
signed long int	4bytes	same as long int
unsigned long int	4bytes	0 to 4,294,967,295
float	4bytes	+/- 3.4e +/- 38 (~7 digits)
double	8bytes	+/- 1.7e +/- 308 (~15 digits)
long double	8 or 10 bytes	+/- 1.7e +/- 308 (~15 digits)
wchar_t	2 or 4 bytes	1 wide character

نکته

- طول (حجم) انواع داده در زبان C یا C++ دقیقاً مشخص نشده است
  - زبان، بازه مقادیر متغیرها را مشخص نکرده است
  - به کامپایلر و سخت‌افزار وابسته است
- مثلاً یک متغیر از نوع int معمولاً در دو یا چهار بایت ذخیره می‌شود
  - اجباری وجود ندارد که حتماً در چهار بایت ذخیره شود
  - بعضی از زبان‌های برنامه‌نویسی (مثل جاوا) بازه مقادیر هر نوع داده را مشخص کرده‌اند
- قاعده:
  - حجم متغیری از جنس short int کمتر یا مساوی int
  - و int هم کمتر/مساوی long int

مثال

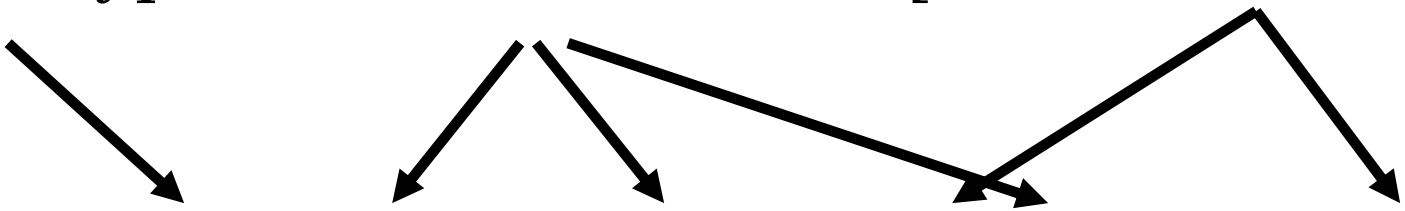
```
int main()
{
 unsigned int age = 32;
 long int big = 2147483647;
 unsigned long int bigger = 4294967295;

 float pi = 3.14;
 double real = 12757.12763173;

 return 0;
}
```

تمرین

- برنامه‌ای بنویسید که شعاع دایره را در متغیری با نام radius نگه دارد
- مساحت دایره را محاسبه کند و در متغیر دیگری نگه دارد

نحوه تعریف متغیر

<variable_type> <variable_name> [=<initial_value>];



- Example: int length, width = 5, height = 10;

انواع خطا

- خطای نحوی (Compiler Error یا Syntax Error)
  - خطایی که کامپایلر آن را کشف می‌کند
  - در اثر رعایت نکردن قواعد نحوی زبان (syntax) ایجاد می‌شود
  - مثلاً: نام متغیر را با رقم شروع کنیم (int 1a;)
- خطای زمان اجرا (Runtime Error)
  - مثلاً انجام تقسیم بر صفر
- خطای منطقی (Logical Error)
  - اشتباه منطقی در برنامه‌نویسی
  - مثال: محاسبه غلط مساحت مستطیل

تشبیه انواع خطاها در [دستورات] زبان طبیعی

- **خطای نحوی**: شما به طبقه سوم بروم!
  - از نظر زبان فارسی این جمله غلط است. قابل اجرا نیست.
- **خطا در زمان اجرا**: به طبقه هفتم ساختمان cse.sbu برو!
  - از نظر زبان فارسی صحیح است. قابل اجرا هست.
  - ولی در اجرا به خطا می‌خوریم: ساختمان cse.sbu طبقه هفتم ندارد!
- **خطای منطقی**: برای رسیدن به کلاس مبانی، به کلاس کنار آمفی تیاتر بروید
  - از نظر نحو زبان صحیح است. قابل اجرا هم هست (می‌توان به کلاس کنار آمفی تیاتر رفت)
  - اما این دستور اشکال منطقی دارد (در این جا اشتباه محاسباتی). کلاس مبانی کنار آمفی تیاتر نیست

مثال

- خطای نحوی (syntax error یا compile error)

```
int age = Ali;
int 1age = 5;
```

- خطا در زمان اجرا (runtime error)

```
int a = 5 - 5 ;
int b = 1/a ;
```

➔ *این دستور خطای زمان اجرا دارد*

- خطای منطقی (logical error)

```
int perimeter = 2 * r * r * 3.14;
```

عملگرها (Operators)

عملگر

- عملگر یا Operator مانند دستگاهی است
  که یک یا چند ورودی (عملوند یا Operand) را دریافت می‌کند
  و پردازشی روی آن‌ها انجام می‌دهد
  و یک نتیجه تولید می‌کند
- مثلاً عملگر جمع (+) دو عملوند می‌گیرد و حاصل‌جمع آن‌دو را برمی‌گرداند
- عملگرهای یک‌عملوندی (unary)
  - مانند – در 1- یا a–
- عملگرهای دو عملوندی (binary)
  - مانند جمع و تفریق
- عملگرهای سه‌عملوندی (ternary)

عملگرهای ریاضی

عملگر ریاضی	نماد	عبارت جبری	مثال
جمع	+	a+5	a + 5
تفریق	–	x-y	x – y
ضرب	*	ax یا a.x	a * x
تقسیم	/	$\frac{x}{y}$ یا x ÷ y	x/y
باقیمانده	%	x mod y	x % y

- عملگرهای فوق همگی دو عملوندی (binary) هستند

مثال

```
int main()
{
 int length, width, perimeter;
 length = 10;
 width = 7;
 perimeter = 2*length + 2*width;

 width = perimeter/2 – length;

 int remainder = perimeter % 2 ;

 return 0;
}
```

نکته درباره عملگرها

- عملگرهای / ,* ,- ,+ را در نظر بگیرید
- برای عملوندهای صحیح (مثل متغیرهای int):
  - خروجی همه آن‌ها اعداد صحیح خواهد بود
  - مثلاً نتیجه 5/2 ، عدد صحیح ۲ است (و نه 2.5)
- برای عملوندهای اعشاری (مثل متغیرهای float):
  - خروجی آن‌ها هم اعشاری است
  - مثلاً نتیجه 1.0/2 ، 0.5 است

عملگر انتساب (Assignment)

- برای مقداردهی به یک متغیر به‌کار می‌رود
- سمت چپ این عملگر «متغیر» و سمت راست آن «مقدار» می‌آید
- بارها از آن استفاده کرده‌ایم
- مثال:

```
int main()
{
 int length, width, perimeter;
 length = 10;
 width = 7;
 perimeter = 2*length + 2*width;

 width = perimeter/2 – length;

 int remainder = perimeter % 2 ;

 return 0;
}
```

تمرین

- فرض کنید متغیرهای a و b در برنامه ما تعریف شده‌اند و مقدار دارند
  - فرض کنید هر دو از نوع int هستند
- قطعه‌برنامه‌ای بنویسید که مقدار a و b را با هم عوض کند
  - مثلاً اگر a برابر با ۵ و b مساوی ۳ باشد، بعد از اجرای کد شما باید a=3 و b=5 بشود

دنبال کردن برنامه (trace)

البته این برنامه به زبان C یا C++ نیست
یک شبه‌کد (pseudocode) است

Algorithm

```
1  number = 3
2  PRINT number
3  FOR  i  from 1 to 3:
4      number = number + 5
5      PRINT number
6  PRINT " ? "
```

Trace Table

Line	number	i	OUTPUT
1	3		
2			3
3		1	
4	8		
5			8
3		2	
4	13		
5			13
3		3	
4	18		
5			18
6			?

- یعنی خودمان را جای کامپیوتر بگذاریم
- دستور به دستور برنامه را دنبال کنیم (در ذهنمان اجرا کنیم)
- در هر مرحله وضعیت برنامه را رصد کنیم
  - مقدار جدید متغیرها را مشخص کنیم (Trace Table)
  - خروجی‌های برنامه را مشخص کنیم

عملگرهای میانبُر

- گاهی اولین عملوند یک عملگر، همان متغیری است که قرار است نتیجه عملگر در آن ذخیره (انتساب) شود
- مثال:

- a = a* 2;
- b = b – 1;

- در این مواقع می‌توانیم از عملگرهای میانبُر انتساب استفاده کنیم:

- *= /= += –=

- مثلاً a+=2 دقیقاً یعنی a=a+2
- و یا b/=2 یعنی b = b/2

```
int length = 5 , width = 2;
length += 2;
length *= length;
length /= width;
length -= width;
```

عملگرهای تک‌عملوندی (Unary Operators)

- عملگر – و +
- مثال : -1 -number +5 +x
  - (عملگر + در واقع کار خاصی انجام نمی‌دهد، گاهی برای تصریح و وضوح برنامه خوب است)
- عملگر ++ : قبل یا بعد از یک متغیر قرار می‌گیرد و مقدار آن را یکی اضافه می‌کند
- مثال : ++number number++
- number++ معادل number+=1 معادل number=number+1
- عملگر -- : قبل یا بعد از یک متغیر قرار می‌گیرد و مقدار آن را یکی کم می‌کند
- مثال : --number number—
- x-- معادل x-=1 معادل x=x-1

عملگرهای ++ و --
Prefix increment and decrement

- عملگرهای محبوب و پرکاربردی هستند
- از نتیجه نهایی این عملگرها در عمل انتساب (assignment) نیز می‌توان استفاده کرد
- مثال ➔ number = age++; age = year--; age= ++x; x=--age;
- نکته مهم درباره ++ و - -
  - اگر قبل از عملوند بیاید، ابتدا اعمال می‌شود و مقدار جدید عملوند در انتساب استفاده می‌شود
  - اگر بعد از عملوند بیاید، ابتدا مقدار عملوند در انتساب استفاده می‌شود و سپس مقدار عملوند عوض می‌شود
- مثال: فرض کنید مقدار x=1 و y=2 باشد. بعد از اجرای هر یک از دستورات زیر مقدار جدید x و y ؟

```
x = y++;
```

```
x = y--;
```

```
x = ++y;
```

```
x = --y;
```

تمرین

```
int main()
{
int age=0, year=1983, remains=32;
age++;
--remains;
age = remains;
age = -- remains;
age = remains--;
age -= 5;
year += 5;
remains = year++ + ++age;

return 0;
}
```

- برنامه زیر را trace کنید

age	year	remains
0	1983	32
1	1983	32
1	1983	31
31	1983	31
30	1983	30
30	1983	29
25	1983	29
25	1988	29
26	1989	2014

اولویت عملگرها (Operator Precedence)

- معنای عبارت زیر چیست؟ ابتدا جمع انجام می‌شود یا ضرب؟

- `age = 2 + 5 * 3;`

- نتیجه: ۱۷ (ابتدا ضرب انجام می‌شود)
- اولویت (precedence) بعضی عملگرها بیشتر است
  - مثلاً اولویت ضرب از جمع بیشتر است
  - اولویت ضرب و تقسیم بیشتر از جمع و تفریق است

ترتیب اولویت عملگرها

Operator	Description
( )	Parentheses
++ --	Prefix/Postfix increment/decrement
+ -	Unary plus/minus
* / %	Multiplication/division/modulus
+ -	Addition/subtraction
=	Assignment

مثال (اولویت عملگرها)

- مقدار متغیر number چه خواهد بود؟

```
int number = 2 + 3 * 4 / 2 ;
```

- پاسخ صحیح: ۸
- البته همواره پرانتزگذاری برای پرهیز از ابهام توصیه می‌شود:

```
int number = 2 + ((3 * 4) / 2) ;
```

- مثال دیگر:

```
z = p * r % q + w / x - y;
```

عملگرها با چه ترتیبی اجرا می‌شوند؟

مثال

```
int age=0, year=1983, remains=32;
age = year = remains++;
year = --remains + age++ * -2;
```

age	year	remains
0	1983	32
32	32	33
33	-32	32

شرکت‌پذیری عملگرها (Associativity)

- وقتی دو عملگر با یک اولویت یکسان در یک عبارت قرار می‌گیرند،
- ارزیابی این عبارت با توجه به خاصیت «شرکت‌پذیری» عملگرها صورت می‌پذیرد
- شرکت‌پذیری: «راست به چپ» یا «چپ به راست»
- مثال: x= y = z = 17 به صورت x = (y = (z = 17)) اجرا می‌شود
  - زیرا شرکت‌پذیری عملگر مقداردهی، راست به چپ است
- مثال: 72 / 2 / 3 به صورت (72 / 2) / 3 اجرا می‌شود
  - زیرا شرکت‌پذیری عملگر تقسیم، چپ به راست است

خلاصه عملگرها

- «اولویت + شرکت‌پذیری» عملگرها، پرانتزهایی که نگذاشتیم را مشخص می‌کنند

Operator	Description	Associativity
++	unary postfix increment	right to left
--	unary postfix decrement	
++	unary prefix increment	right to left
--	unary prefix decrement	
+	unary plus	
-	unary minus	
*	multiplication	left to right
/	division	
%	remainder	
+	addition or string concatenation	left to right
-	subtraction	

پرانتزگذاری

- از پرانتزگذاری مناسب برای پرهیز از ابهام استفاده کنید
  - (پرهیز از ابهام برای خودتان و برنامه‌نویسان، نه برای کامپیوتر)
  - تا برنامه شما راحت‌تر و روان‌تر خوانده و فهمیده شود

```
age = (--remains) + (age++ * -2);
```

```
int months = (2015 - year) * 12;
```

عبارت (expression) و ارزیابی (evaluation)

- عبارت (expression): یک ترکیب معتبر از علایم که در نهایت یک مقدار را نشان می‌دهد
- مثال:

- 5
- 5+2
- a+5
- ((++a) * y) – 7
- a * y – 7

- هر عبارت، باید ارزیابی (evaluate) شود تا مقدار آن مشخص شود
  - در آخرین مثال فوق:
  - ابتدا * و سپس - ارزیابی می‌شوند تا مقدار نهایی محاسبه شود

عملگر سه‌عملوندی شرطی

- نحوه استفاده:

```
TYPE value = CONDITION ? Val_True : Val_False;
```

- مثال:

```
double area = 2 * 2 * 3.14;
int value = area > 10 ? 1 : -1;
//value = 1;
```

```
if(area>10)
    value = 1;
else
    value = -1;
```

لیترال‌ها (Literal)

- هر نوع داده شامل مقادیر مختلفی است
- هر مقدار، یک لیترال (literal) خوانده می‌شود
  - مثلاً 12 یک لیترال از نوع عدد صحیح است و 3.14 یک لیترال از نوع عدد اعشاری
- لیترال‌های هر نوع داده به شکلی نمایش داده می‌شوند

- Integer literals: ***0, 12, -7, 256, 0xa3*** (0xa3 = 163)
- Floating point literals: ***3.14, -5.0, .7, -0.54, 1.2e3*** (1.2e3 = 1200.0)
- Character literals: ***'x', 'A', '7', '@', '\n'***
- Boolean literals: ***true, false***
- String literals: ***"Ali Alavi", "This is a test!"***

لیترال کاراکتر

- یک لیترال از نوع کاراکتر، بین دو کوتیشن (') محصور شده است
- نگهداری کاراکترها با کد اسکی آن‌ها انجام می‌شود
  - مثلاً به ازای دستور char ch = 'A'; مقدار ۶۵ در حافظه‌ی متغیر ch ذخیره می‌شود
  - (کد اسکی کاراکتر A ، عدد ۶۵ است)
- بعضی از کارکترهای خاص از ترکیب \ و یک سمبل مشخص می‌شوند
  - مثلاً '\n' یعنی کاراکتر new line
  - و یا '\t' یعنی کاراکتر tab
- مثلاً نتیجه اجرای printf("Hello\tWorld\nYes"); چیست؟

```
Hello        World
Yes
```

An escape sequence starts with a ‘\’ (backslash)

کاراکترهای خاص

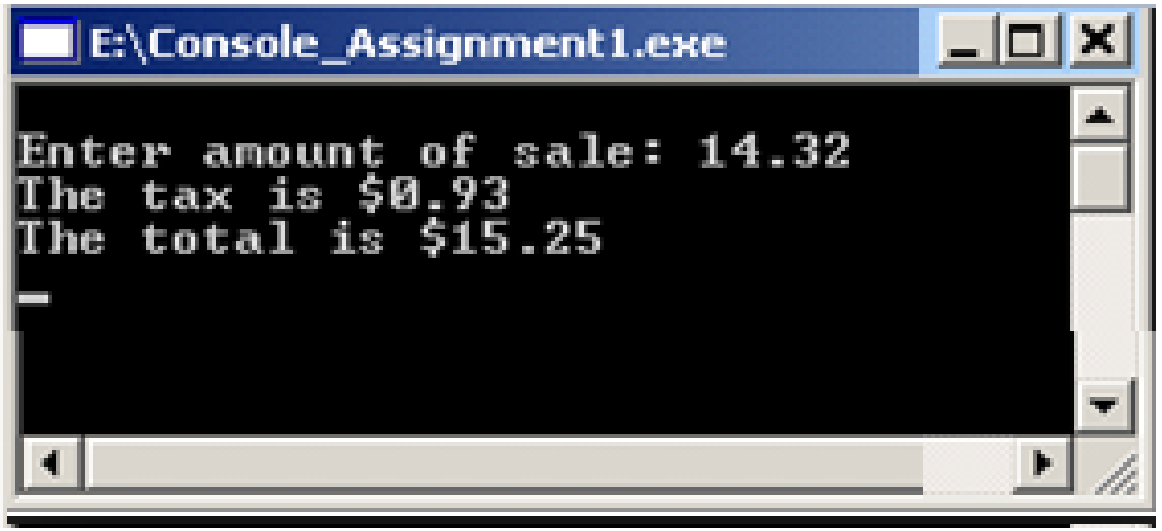
Escape Sequence	Description
\'	single quote
\"	double quote
\\	backslash
\b	backspace
\n	new line
\r	carriage return
\t	horizontal tab

ورودی و خروجی استاندارد

توابع ورودی و خروجی

- برنامه‌هایی که ما (فعلاً) می‌نویسیم مبتنی بر کنسول (Console Application) هستند
  - سایر شکلهای برنامه‌ها؟
    - Web app, Mobile app, GUI Desktop app
- برنامه‌های مبتنی بر کنسول دارای ورودی و خروجی استاندارد هستند
  - Standard Input & Standard Output
- خروجی استاندارد: برنامه از این طریق خروجی را به کاربر نمایش می‌دهد
  - برای این کار از تابع printf استفاده می‌کنیم
- ورودی استاندارد: کاربر از این طریق ورودی را به برنامه وارد می‌کند
  - برای این کار از تابع scanf استفاده می‌کنیم

برنامه‌های مبتنی بر کنسول




```
Welcome to Product Database Application
_____________________________________________

1. Add Product
2. Edit Product
3. View Product
4. Delete Product
5. Exit

_____________________________________________

Enter your Selection:
```

نمایش خروجی با کمک printf

- مقادیر موردنظر را نمایش می‌دهد
- این دستور (تابع) یک یا چند پارامتر (آرگومان) دریافت می‌کند
- پارامتر اول: یک رشته‌ی کنترلی است که چگونگی نمایش مقادیر را مشخص می‌کند
  - کاراکترهای رشته‌ی کنترلی عیناً چاپ می‌شوند، مگر در مواردی که با % همراه شده‌اند
  - به ازای هر پارامتر دستور printf ، باید یک جایگاه با کمک % در رشته کنترلی مشخص شود (البته به جز پارامتر اول که خودش رشته کنترلی است)
- مثال:

```
int value = 5;
printf("%d lt %d", value, 6);
```

خروجی:

```
5 lt 6
```

کاراکترهای کنترلی در توابع printf و scanf

مفهوم	کاراکتر کنترلی
یک عدد صحیح در مبنای ده	%d
یک عدد اعشاری float	%f
یک عدد اعشاری double	%lf
یک کاراکتر	%c
کاراکتر %	%%

مثال

```c
int a = 5;
printf("a=%d\n", a);
float pi = 3.14;
printf("pi is equal to %f\n", pi);
char ch = 'A';
printf("ch=%c\n", ch);

printf("%d , %f , %c , 100%% \n", a, pi, ch);
```

خروجی:

```
a=5
pi is equal to 3.140000
ch=A
5 , 3.140000 , A , 100%
```

تابع scanf

- برای گرفتن مقدار متغیر از ورودی
- پارامتر اول scanf (مشابه printf)

یک رشته کنترلی: مشخص‌کننده نوع متغیرهایی که قرار است خوانده شوند

- پارامترهای بعدی این متغیرها را مشخص می‌کنند (متغیرهایی که قرار است خوانده شوند)
- نکته: هر متغیر باید با یک & همراه شود
- مثلاً

- scanf("%d", &age);

- مقدار متغیر age را به عنوان یک عدد صحیح از ورودی دریافت می‌کند

مثال

```
printf("Please enter the radius:");
float radius;
scanf("%f", &radius);
float area = 3.14 * radius * radius;
printf("MASAHAT= %f", area);
```

- (فرض کنید مقدار ۲ به عنوان ورودی وارد شود)

خروجی:

```
Please enter the radius:2
MASAHAT = 12.560000
```

چند نکته دیگر درباره متغیرها

متغیرهایی که مقداردهی نشده‌اند

- اگر یک متغیر را بدون مقداردهی استفاده کنیم، مقداری نامشخص خواهد داشت
- مثال:

```
int m;
printf("%d", m);
```

خروجی:

```
2130567168
```

- البته این نکته درباره متغیرهای «محلی» صادق است
  - متغیرهای global و متغیرهای static با صفر مقداردهی می‌شوند
  - بعدها با این نوع متغیرها آشنا خواهیم شد

ثابت‌ها (constant)

- یک ثابت، متغیری است که تنها یک بار مقداردهی می‌شود
  و پس از آن تغییر دادن مقدار آن در ادامه‌ی برنامه ممکن نیست
- تعریف ثابت: کلیدواژه‌ی const به ابتدای تعریف متغیر اضافه می‌شود
  (قبل از بیان نوع داده)

```
const double PI = 3.141592;
const double E =  2.71828;
const char NEW_LINE = '\n';
const int LONG_INT = 2147483647;

PI = 2;  // ➔ Syntax Error
```

جمع‌بندی

جمع‌بندی

- مفهوم متغیر
- انواع داده
- عملگرها
  - اولویت عملگرها
- ورودی و خروجی استاندارد
  - printf و scanf

مطالعه

- فصل ۲ از کتاب C How to Program (7^{th} edition)

Chapter 2 : Introduction to C Programming

- و یا فصل ۲ از کتاب C++ How to Program (8^{th} edition)

Chapter 2 : Introduction to C++ Programming

(در این مرجع، cin و cout به جای printf و scanf معرفی شده‌اند)

مطالعه و جستجوی بیشتر

- عرف نامگذاری (Naming Convention) در زبان C و C++
  - ذیل مبحث coding standards و یا coding conventions جای می‌گیرد
- آیا زبان C و C++ ، وابسته به محیط خاصی هستند؟
  - مثلاً به سیستم‌عامل یا سخت‌افزار خاصی وابسته هستند؟
  - اصطلاح Platform Independence
  - آیا C و C++ در عمل هم Platform Independence هستند؟
    - چرا؟ (مثلاً درباره طول متغیرها در حافظه)
  - چه زبان‌هایی Platform Independence هستند؟

مطالعه و جستجوی بیشتر (ادامه)

- کاراکترهای ’\r‘ و ’\n‘ چه تفاوتی با هم دارند؟
  - فایل‌های متنی که در ویندوز ساخته می‌شوند، چه تفاوتی با فایلهای متنی که در لینوکس ساخته می‌شوند دارند؟
- چگونه می‌توانیم ورودی و خروجی استاندارد را تغییر دهیم؟
  - مثلاً به جای کنسول، خروجی‌ها در یک فایل ذخیره شوند؟
- اگر تعداد پارامترهای تابع printf یا scanf با رشته کنترلی (پارامتر اول) مطابق نباشد، خطای کامپایل رخ می‌دهد یا خطای زمان اجرا؟ چرا؟
- چه کاراکترهای کنترلی دیگری برای توابع printf و scanf وجود دارد؟
  - مثلاً %s و %x و %u یعنی چه؟
- قالب رشته‌ای (formatted string) که در printf و scanf استفاده می‌شود، در چه توابع دیگری استفاده می‌شود؟ حتی در کدام زبان‌های برنامه‌نویسی دیگر؟ مثلاً چگونه در جاوا؟

مطالعه و جستجوی بیشتر (ادامه)

- خروجی printf گاهی بلافاصله چاپ نمی‌شود. چرا؟
  - راهنمایی: Buffered output
- معنای این دستور چیست؟ printf("area = %7.2f\n", area);

مطالب تکمیلی

نکته

- اولویت و شرکت‌پذیری ترتیب اجرای عملگرها را مشخص می‌کنند، نه عملوندها
- عملوندها به هر ترتیبی ممکن است evaluate شوند

Order of Evaluation

- Order of evaluation of the operands of almost all C++ operators is unspecified.
  - (including the order of evaluation of function arguments in a function-call expression and the order of evaluation of the subexpressions within any expression)
- The compiler can evaluate operands in any order, and may choose another order when the same expression is evaluated again.
- This is not to be confused with left-to-right and right-to-left associativity of operators:
  - the expression f1() / f2() * f3() is parsed as (f1() / f2()) * f3() due to left-to-right associativity of operator * and /,
  - but the function call to f3 may be evaluated first, last, or between f1() or f2() at run time.

- http://en.cppreference.com/w/cpp/language/eval_order

وقتی نتیجه کار مشخص نیست

- گاهی (برای برخی دستورهایی که می‌نویسیم) رفتار زبان «تعریف‌نشده» است
- رفتار کامپایلرهای مختلف ممکن است متفاوت باشد
  - حتی رفتار یک کامپایلر در شرایط مختلف ممکن است متفاوت باشد

```
int main() {
int age=32;
age = age++;

return 0;
}
```

Undefined behavior

```
int main() {
int age=32;
age = age++ + age++ + age++;

return 0;
}
```

Undefined behavior

وقتی نتیجه کار مشخص نیست (ادامه)

```
int main() {
int age=25, year=1988, remains=29;
year = year++ + ++age;

return 0;
}
```

Undefined behavior

```
int age=32, year=32, remains=33;
age = --remains + age++ * -2;
```

Undefined behavior

More about Literals

- An integer literal can be a decimal, octal, or hexadecimal constant
  - A prefix specifies the base or radix: 0x or 0X for hexadecimal
  - 0 for octal, and nothing for decimal.
- An integer literal can also have a suffix that is a combination of U and L, for unsigned and long, respectively.
  - The suffix can be uppercase or lowercase and can be in any order.

```
212         /* Legal */
215u        /* Legal */
0xFeeL      /* Legal */
078         /* Illegal: 8 is not an octal digit */
032UU       /* Illegal: cannot repeat a suffix */
```

```
85          /* decimal */
0213        /* octal */
0x4b        /* hexadecimal */
30          /* int */
30u         /* unsigned int */
30l         /* long */
30ul        /* unsigned long */
```

Complicated Formatted Strings

- An example of a format string is
- "%7d%s %c%lf"
- The above format string scans the first seven characters as a decimal integer, then reads the remaining as a string until a space, new line or tab is found, then scans the first non-whitespace character following and a double-precision floating-point number afterwards.

پایان