فصل اول
عملگرها و عملیات‌ها

۱-۱ عملیات حسابی

در زبان C++، ۴ عمل اصلی شناخته‌شده هستند یعنی می‌توانیم ۲ عدد را با علامت + با هم جمع کنیم. هم‌چنین، برای ضرب از * ، برای تفریق از - و برای تقسیم از / استفاده می‌کنیم. هم‌چنین، برای گرفتن باقی‌مانده‌ی a بر b از ٪ استفاده می‌کنیم. مثلاً وقتی می‌نویسیم:

```
a = 3 * 5 - 1;
```

ابتدا مقدار ۵×۳ حساب می‌شود سپس یک واحد از آن کم می‌شود و عدد نهایی درون a ریخته می‌شود. دقت کنید که کامپیوتر اولویت‌ها را می‌شناسد و طبق آن‌ها عمل می‌کند یعنی در مثال بالا ابتدا ضرب و سپس تفریق را انجام می‌دهد. اگر می‌خواستیم که اول تفریق انجام شود باید با پرانتز آن را اولویت می‌دادیم؛ یعنی:

```
a = 3 * (5 - 1);
```

هم‌چنین، = به معنی قرار دادن مقدار متغیر سمت راستش در سمت چپش است. برای مثال:

```
a = b;
```

یعنی مقدار داخل متغیر b را داخل a قرار بدهد؛ بنابراین، بعد از این دستور مقدار داخل متغیر a برابر مقدار داخل متغیر b است. دقت کنید = به هیچ عنوان تساوی دو طرف را چک نمی‌کند و برای چک کردن تساوی دو متغیر از نماد دیگری استفاده می‌کنیم که با این اشتباه نشود. برای مثال:

```
a = 3;
b = 3 * 6;
c = b * 2;
b = b + c;
c = c + 1;
a = c / 2;
```

در خط اول = وجود دارد، پس مقدار سمت راست آن داخل مقدار سمت چپ ریخته می‌شود؛ یعنی عدد ۳ داخل a ریخته می‌شود و از این پس مقدار آن ۳ است.

در خط دوم، مجدد = وجود دارد، پس باید مقدار سمت راست محاسبه‌شده و در سمت چپ ریخته شود. برای این کار، مقدار سمت راست، خود یک ضرب دارد و ابتدا باید حساب شود، پس کامپیوتر ابتدا ۶×۳ را محاسبه می‌کند و به‌جای آن عدد ۱۸ را سمت راست عبارت جایگزین می‌کند، پس از آن، ۱۸ را داخل b می‌ریزد.

در خط بعد = وجود دارد و سمت راست × داریم، پس ابتدا مقدار b در ۲ ضرب می‌شود یعنی۳۶=۲×۱۸ و عدد ۳۶ داخل c ریخته می‌شود. در خط بعد، ابتدا سمت راست محاسبه می‌شود یعنی مقادیر b و c باهم جمع می‌شوند، پس ۵۴=۳۶+۱۸ و حالا مقدار ۵۴ داخل b ریخته می‌شود. از این پس دیگر مقدار قبلی b از بین می‌رود و داخل آن ۵۴ وجود دارد. این نکته حائز اهمیت است که وقتی داخل b مقداری ریخته می‌شود، دیگر مقدار قبلی‌اش کاملاً از دست می‌رود. در خط بعد، مقدار c+۱ یعنی ۳۷=۱+۳۶ حساب می‌شود و داخل c ریخته می‌شود پس دیگر مقدار ۳۷ را دارد و مقدار قبلی‌اش یعنی ۳۶ از بین رفت. این خط هم خط مهمی است، زیرا برای یک واحد اضافه کردن یک متغیر از این دستور استفاده می‌کنیم. هم‌چنین، برای یک واحد کم یا زیاد کردن متغیری مثلاً d می‌نویسیم:

```
d--;d++;
```

در خط بعد هم مقدار c که ۳۷ است تقسیم بر ۲ می‌شود ۱۸/۵=۲÷۳۷ که چون a باید عدد صحیح باشد مقدار ۱۸ توسط خود کامپیوتر در نظر گرفته می‌شود و داخل a ریخته می‌شود. پس در آخر، مقدار a برابر ۱۸، مقدار b برابر ۵۴ و مقدار c برابر ۳۷ است.

خیلی اوقات لازم می‌شود که متغیری را زیاد، کم و یا ضرب کنیم. برای این کار، به‌طور مثال یک واحد اضافه کردن متغیر a، می‌نویسیم:

```
a = a + 1;
```

در زبان C++، برای این کار مخففی وجود دارد. هم‌چنین، باعث خوانایی برنامه خواهد شد. این دستور به شکل زیر است:

```
a += 1;
```

و به‌طورکلی، برای ضرب کردن a در b می‌نویسیم:

```
a *= b
```

که دقیقاً مشابه:

```
a = a * b;
```

است.

۲-۱ عملیات منطقی

علاوه بر عملگرهای حسابی، عملگرهای منطقی هم موجود است که معمولاً برای چک کردن به کار می‌روند. علامت کوچک‌تر به شکل >، کوچک‌تر مساوی به شکل =>، بزرگ‌تر به شکل< و بزرگ‌تر مساوی به شکل =< است. مثلاً برای این‌که بفهمیم مقدار b بزرگ‌تر از مقدار a هست یا خیر، می‌نویسیم b>a. وقتی کامپیوتر به b>a می‌رسد، بررسی می‌کند که آیا

مقدار b از مقدار داخل a بزرگ‌تر هست یا خیر، اگر بزرگ‌تر باشد به معنی آن هست که b>a درست است پس به جای کل این عبارت مقدار true یا همان ۱ گذاشته می‌شود.

بنابراین، زمانی که به‌عنوان یک شرط بیان‌شده باشد، ابتدا مقدار آن محاسبه می‌شود و اگر صحیح بود با true یا ۱ و اگر صحیح نبود با ۰ یا false جایگزین می‌شود (توسط خود کامپیوتر) و از آن پس اگر حاصل ۱ بود به معنی صحیح بودن شرط، کارهای آتی را انجام خواهد داد.

یکی دیگر از عملگرهای مقایسه‌ای == است که برای برابر بودن دو متغیر استفاده می‌شود. وقتی می‌نویسیم a==b، یعنی آیا مقدار a برابر مقدار b هست یا خیر، اگر برابر بود، این عبارت صحیح خواهد شد و در غیر این صورت، این عبارت false خواهد شد. همان‌طور که در بالا اشاره شد، = به معنی ریختن مقدار متغیر سمت راست در متغیر سمت چپ است و == به معنی چک کردن تساوی دو متغیر. دقت شود که این دو، جای هم به کار نروند وگرنه باعث درست کار نکردن برنامه خواهد شد. هم‌چنین از =! برای نابرابر بودن استفاده می‌شود. مثلاً وقتی داریم a=!b یعنی اگر b با a برابر نبود، یا به عبارتی وقتی این عبارت صحیح است که a با b برابر نباشد و در غیر این صورت این عبارت ۰ خواهد شد. توجه کنید که حاصل هر عبارت مقایسه‌ای، در نهایت ۰ یا ۱ خواهد شد که کامپیوتر خودش پس از چک کردن برقراری رابطه اگر صحیح بود کل آن عبارت را با ۱ و در غیر این صورت با ۰ جایگزین می‌کند.

۱-۳ سؤالات

۱. مقادیر متغیرها را در پایان برنامه به دست آورید.

```
b = 8;
a = 4 + 1 * 2;
b == 2;
a = 4 - 1 / 2;
c = a + 1;
c = c + 1;
d = c = a;
e = c == 5;
a == a = b;
a = b <= c;
a < b < c;
```

۴-۱ راه‌حل‌ها

۱. در ابتدا ۸ داخل متغیر b ریخته می‌شود.

سپس در خط بعد سمت راست آن باید حساب شود؛ یعنی ابتدا ۲×۱ حساب می‌شود.

سپس حاصل آن با ۴ جمع می‌شود.

سپس حاصل آن یعنی ۶ در a ریخته می‌شود.

دقت کنید که ترتیب‌ها بسیار مهم هستند و کامپیوتر ابتدا سمت راست را کامل حساب می‌کند و در هر مرحله مقدار محاسبه‌شده را جایگزین می‌کند. برای مثال وقتی ۲×۱ حساب شد و مقدار آن ۲ شد ۲×۱ کامل پاک می‌شود و جای آن ۲ گذاشته می‌شود. حال ۲+۴ حساب می‌شود و قبل از اینکه در a ریخته شود مقدار پاک‌شده و جایگزین می‌شود. سپس سمت راست ۶ و سمت چپ a هست پس ۶ را داخل a می‌ریزد.

در خط بعد چون دو تا مساوی داریم چک می‌شود که آیا b برابر ۲ هست یا نه که چون b، ۸ است و مساوی نیست پس کل خط b==۲ با ۰ جایگزین شده بنابراین، این خط هیچ فایده‌ای ندارد و مقدار b ثابت می‌ماند.

در خط بعد ۱÷۲ می‌شود ۰/۵ اما در کامپیوتر چون پیش‌فرض همیشه عدد صحیح بودن هست چون مقدار آن می‌شود ۰/۵ که عدد صحیح نیست، قسمت صحیح آن یعنی ۰ در نظر گرفته می‌شود و ۰ جایش قرار می‌گیرد.

سپس ۰−۴ حساب‌شده و ۴ جایگزین می‌شود و پس از آن مقدار ۴ در a ریخته می‌شود. پس الان در a، ۴ وجود دارد و دیگر مقدار قبلی خود را از دست داده است. در حقیقت ۴ روی قبلی ریخته شد.

سپس در c مقدار a که ۴ هست با ۱ جمع می‌شود یعنی ۵ و در c ریخته می‌شود. دقت کنید که مقدار a ثابت می‌ماند و با ۱ جمع نمی‌شود.

سپس مقدار c یعنی ۵ با ۱ جمع می‌شود و در خودش یعنی c ریخته می‌شود، یعنی c برابر ۶ می‌شود. دقت شود برای این‌که یک متغیر ۱ واحد افزایش پیدا کند این کار را می‌کنیم.

سپس d=c=a اگر از سمت راست کارها را یکی‌یکی انجام دهیم، ابتدا a داخل c ریخته می‌شود یعنی همان ۴ پس c، ۴ می‌شود و سپس در d، ۴ ریخته می‌شود. پس این خط هر ۳ متغیر را برابر a یعنی همان ۴ می‌کند و از اینجا به بعد هر متغیر مقدار ۴ را دارد.

در خط بعد، از سمت راست، ابتدا c==۵ بودن چک شده که چون == است یعنی چک شدن، پس جواب آن یا یک هست یا صفر که چون c برابر ۴ است اما سمت راست ۵ است پس این مساوی برقرار نیست و کل عبارت با ۰ جایگزین می‌شود، حال ۰ در e ریخته می‌شود.

در خط بعد، سمت راست == طبق تعریف، باید یک متغیر باشد، نه یک عبارت، پس این خط error خواهد داد و برنامه اجرا نخواهد شد (امتحان کنید و با تصحیح آن، برنامه را اجرا کنید).

در خط بعد، b یعنی عدد ۸ چک می‌شود که از c کوچک‌تر یا مساوی باشد که چون c، ۴ است پس این شرط برقرار نیست و جایش ۰ می‌آید. سپس در a، ۰ ریخته می‌شود.

در خط بعد هم که ابتدا c>b بودن چک می‌شود، یعنی ۴>۸ که چون برقرار نیست جای آن ۰ می‌آید و سپس a<۰ بودن چک می‌شود که باز هم برقرار نیست. دقت کنید که متوجه می‌شویم که این خط کلاً اتفاق نادرست و دور از انتظاری می‌افتد و اصلاً مثل ریاضی نیست که این خط وقتی درست باشد که هم a کوچک‌تر از b باشد هم b کوچک‌تر از c.

فصل دوم
انواع داده‌ها

برای گرفتن حافظه توسط کامپیوتر، لازم است که نوع حافظه را مشخص کنیم. فرض کنید می‌خواهیم یک عدد صحیح را در کامپیوتر ذخیره کنیم؛ بنابراین، باید به کامپیوتر بگوییم که یک حافظه از نوع عدد صحیح به ما بدهد و پس از آن، هم ما و هم کامپیوتر بداند که با داده‌ی داخل آن حافظه چطور برخورد کند. از مهم‌ترین انواع داده‌هایی که وجود دارند عبارت است از:

`int` (`integer`): این نوع داده برای نگهداری عدد صحیح است و حافظه آن ۴ بایت است و از آن‌جایی که هر بایت ۸ بیت خانه دارد که هرکدام ۰ یا ۱ می‌پذیرند، پس جمعاً ۳۲=۴×۸ بیت (خانه حافظه) رزرو می‌شود و چون محدوده اعداد مثبت و منفی برای `int` قابل‌قبول است، پس از $-۲^{۳۱}$ تا $۲^{۳۱}-۱$ محدوده عدد قابل‌قبول برای `int` است. برای مثال اگر عددی بزرگ‌تر از محدوده مورد نظر به یک حافظه `int` داده شود، ازآن‌جایی‌که حافظه لازم برای آن وجود ندارد، کامپیوتر به‌صورت خودکار روی آن تغییراتی ایجاد می‌کند و عدد، هنگام ذخیره به‌کلی عوض خواهد شد. پس باید حواسمان باشد که محدودیت‌های هر تایپ را رعایت کنیم.

`short int`: این نوع حافظه ۲ بایت یا همان ۱۶ بیت است و حافظه لازم برای نگهداری آن تنها ۱۶ بیت است؛ یعنی برای نگهداری اعدادی که کوچک هستند بهتر است از `short int` استفاده کنیم. محدوده آن از $-۲^{۱۵}$ تا $۲^{۱۵}-۱$ است.

`long long int`: این نوع داده ۸ بایت یا همان ۶۴ بیت است و محدوده آن از $-۲^{۶۳}$ تا $۲^{۶۳}-۱$ است.

`float`: این نوع داده برای نگهداری اعداد اعشاری است. مقداری بیت به نگهداری قسمت صحیح عدد اختصاص می‌دهد و مقداری بیت هم به قسمت اعشاری عدد اختصاص می‌دهد. به علت کم اهمیت بودن، محدوده دقیق اعداد قابل‌قبول اینجا آورده نشده و در صورت تمایل قابل جستجو در اینترنت است.

`double`: `double` هم مانند `float` مخصوص اعداد اعشاری است با این تفاوت که مقدار حافظه بیشتری می‌گیرد و در عوض اعدادی که می‌تواند ذخیره کند در محدوده بیشتری هستند.

`char` (`character`): این نوع داده مخصوص نگهداری یک کاراکتر است و حافظه آن ۱ بایت است. کاراکترها انواع مختلفی دارند و در حقیقت تمام شکلک‌هایی که در کامپیوتر مشاهده می‌شود هر کدام یک کاراکتر هستند مانند حروف کوچک انگلیسی یا حروف بزرگ انگلیسی و همچنین اعداد ۰ تا ۹ و حتی شکل‌هایی نظیر: * & ^ % و .... در حقیقت هر سمبلی که در کامپیوتر قابل مشاهده است به عنوان یک کاراکتر شناخته می‌شود و قابل ذخیره هست. برای هر کاراکتر، عددی استاندارد کرده‌اند که به‌جای آن، عدد نظیرش را خود کامپیوتر در حافظه قرار می‌دهد و هنگامی‌که به آن خانه از حافظه نگاه می‌اندازیم بازهم خود کامپیوتر طبق جدولش آن عدد را تبدیل به شکل متناظرش می‌کند و آن را نمایش می‌دهد. به استانداردی که وجود دارد و برای هر سمبل یک عدد اختصاص داده‌اند، کد اسکی می‌گویند. در حقیقت طبق جدول زیر، هر سمبلی یک عدد دارد. برای مثال `'A'` عدد ۶۵ است به این معنی که اگر بخواهیم در حافظه‌ای `'A'` را ذخیره کنیم ما می‌نویسیم `'A'` اما در حقیقت خود کامپیوتر بجای `'A'` عدد ۶۵ را ذخیره می‌کند.

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

`bool`: در این نوع داده، فقط دو نوع حالت قابل نگهداری است و مخصوص وضعیتی است که ما می‌خواهیم صحیح یا غلط بودن چیزی را نگهداری کنیم، یا به‌طور کلی می‌خواهیم متغیری را در آن نگاه داریم که آن متغیر دو حالت بیشتر ندارد و آن ۰ و ۱ است. می‌توانیم درون آن‌ها `false` یا `true` بریزیم که `false` معادل ۰ و `true` معادل ۱ است. اگر در متغیر `bool` یک عدد غیر از ۰ و ۱ بریزیم، چون‌که هر عدد غیر صفری در کامپیوتر درست در نظر گرفته می‌شود پس به‌جای آن ۱ ریخته می‌شود که همان `true` است.

متغیرهای دیگری هم وجود دارد از جمله `unsigned int` که مفهوم آن عدد صحیح بدون علامت است که قابل جستجو در اینترنت است و به علت کم اهمیت بودن، اینجا توضیحی در مورد آن‌ها داده نشده است.

۱-۲ سؤالات

۱. در کد زیر چند بایت و چند بیت حافظه گرفته‌شده است؟

```
int a;
int b;
long long int c, d;
char e;
a = 5;
b = a + 1;
```

۲. مقادیر a، b، c و d را در زیر مشخص کنید.

```
int a = 100;
int b = a + 800;
long long int c = -18;
a = b + c;
char d = 's';
char e = 'r';
e = 65;
```

۳. مقدار متغیرهای زیر را در انتهای برنامه بنویسید.

```
int a = 'c';
int b = 100000000000;
long long int c = 100000000000;
char d = 'a' + 2;
char e = 'a' + '2';
e = 35 + 30;
int f = d;
float g = 8.8;
int h = g;
bool k = true;
bool z = 5;
```

```
bool m = 0;
bool n = 1 - m;
char p = n;
```

۲-۲ راه‌حل‌ها

۱. می‌دانیم که هر بایت حافظه شامل ۸ بیت خانه حافظه است. هر بیت هم یک خانه حافظه است که یا ۰ است یا ۱.

می‌دانیم هر int، ۴ بایت است، هر long long int، ۸ بایت و هر کاراکتر ۱ بایت. پس ۲ تا ۴ بایت داریم. چون ۲ تا long long int به نام‌های c و d تعریف‌شده، ۲ تا ۸ بایت هم داریم و در انتها یک ۱ بایت داریم. در خط آخر چون فقط عملیاتی انجام‌شده است و حافظه‌ای تعریف‌نشده است پس حافظه‌ای اضافه‌تر گرفته نمی‌شود. پس کلاً ۲۵ بایت حافظه یا ۲۰۰ بیت حافظه گرفته‌شده است. (حال بهتر درک می‌کنیم فایل متنی که مثلاً حجمش ۳۰۰ کیلوبایت است منظور چه مقدار حافظه است.)

۲. ابتدا متغیر a تعریف‌شده و مقدار آن ۱۰۰ گذاشته‌شده است. سپس متغیر b تعریف‌شده و مقدار آن a+۸۰۰ گذاشته‌شده که چون a از خط قبل مقدار ۱۰۰ را دارد پس a+۸۰۰ توسط کامپیوتر محاسبه می‌شود و مقدار ۹۰۰ داخل b ریخته می‌شود. دقت کنید که مقدار ۹۰۰ ریخته می‌شود و دیگر کاری به a ندارد و اگر a عوض شود هم باز مقدار b، ۹۰۰ می‌ماند.

در خط بعد c را تعریف و مقدار c را ۱۸- گذاشتیم. در خط بعد مقدار a را برابر c+b گذاشتیم که b=۹۰۰ و c برابر ۱۸- است. پس مقدار a که ۱۰۰ بود، ۸۸۲ می‌شود و دیگر مقدار قبلی‌اش از بین می‌رود.

در خط بعد کاراکتر d تعریف‌شده و کاراکتر s در آن ریخته شده است. دقت کنید که در حقیقت کد اسکی s در آن ریخته می‌شود. توجه کنیم که برای ریختن کاراکتری در متغیر باید آن کاراکتر داخل کوتیشن ' ' باشد. در خط بعد حافظه e از نوع کاراکتر تعریف‌شده و کاراکتر r در آن ریخته شده است (کد اسکی آن). در خط بعد داخل e که از نوع کاراکتر است نه عدد (!)، عدد ۶۵ ریخته شده است؛ اما این کار قابل انجام است و کامپیوتر کاری ندارد به e و فکر می‌کند که ما داریم کد اسکی داخل آن می‌ریزیم؛ اما وقتی آن را چاپ می‌کنیم یا کاری انجام می‌دهیم کامپیوتر خودش کاراکتری را پیدا می‌کند که کد اسکی آن ۶۵ است یعنی 'A'. پس در نهایت کاراکتر r پاک می‌شود و A ریخته می‌شود.

۳. خط اول مقدار a برابر کد اسکی c یعنی ۹۹ است.

سپس در خط بعد داخل متغیر از نوع int به نام b عدد مشاهده شده ریخته می‌شود. دقت کنید که int، ۴ بایت یعنی ۳۲ بیت است پس امکان ذخیره در آن از یک عددی بزرگ‌تر وجود ندارد و در صورت این کار، عددی دیگر ذخیره خواهد شد. پس در c یک عددی ذخیره می‌شود اما عدد داخل b از بین خواهد رفت و عدد دیگری در آن خواهد ماند زیرا b از نوع int است. در صورت علاقه می‌توانید مباحث one's complement و two's complement را در اینترنت مطالعه کنید تا طریقه نگهداری اعداد توسط کامپیوتر را دقیق‌تر متوجه بشوید.

در خط بعد کاراکتر a با ۲ جمع شده و داخل d ریخته می‌شود. در حقیقت کاراکتر 'a'، کد اسکی آن است که ۹۷ است و وقتی با ۲ جمع شود ۹۹ می‌شود و داخل d ریخته می‌شود؛ و چون d کاراکتر است، کامپیوتر می‌داند که کد اسکی است و کاراکتر آن یعنی 'c' را به ما نشان می‌دهد. در واقع، کاراکترها، کد اسکی‌شان پشت سر هم هستند پس وقتی به a، ۲ واحد اضافه می‌کنیم 'c' می‌شود.

در خط بعد، داخل کاراکتر e، کاراکتر ۲ و کاراکتر 'a۱' جمع شده، ریخته می‌شود. دقت کنید که اینجا ۲، داخل کوتیشن است یعنی منظور کد اسکی ۲ بوده نه عدد ۲. پس اینجا کد اسکی '۲۱' با کد اسکی 'a۱' جمع می‌شود و یک کاراکتری می‌شود که به داخل e می‌رود. در خط بعد، دو عدد با هم جمع می‌شوند یعنی ۳۵ و ۳۰ و داخل e ریخته می‌شوند که همان ۶۵ یا 'A۱' است.

در f که از نوع int است مقدار d که از نوع کاراکتر است ریخته می‌شود. در حقیقت با این کار، کد اسکی d در f ریخته می‌شود؛ یعنی ۹۹ و این‌بار کامپیوتر با f یعنی ۹۹ مثل int برخورد می‌کند. سپس داخل g مقدار ۸/۸ ریخته شده است. در h که int است مقدار g یعنی ۸/۸ ریخته می‌شود اما چون h، int است، ۰/۸ آن میفتد و ۸ ذخیره می‌شود.

در خط بعد، داخل k مقدار true یا همان ۱ ذخیره می‌شود. در خط بعد داخل z مقدار ۵ ریخته می‌شود که چون bool است و فقط صحیح و غلط یا همان ۰ و ۱ ذخیره می‌شود و چون هر چیزی به‌جز ۰ صحیح است پس ۵ هم صحیح است پس داخل z صحیح یا همان ۱ ریخته می‌شود و ۵ از بین خواهد رفت. خط بعد داخل m، false یا همان ۰ می‌رود. در n مقدار m-۱ یعنی ۱ که همان true است می‌رود پس n، true می‌شود. در حقیقت با عمل h=۱-h، مقدار h برعکس می‌شود (اگر h، bool باشد) و این یک راه برای برعکس کردن یک متغیر بدون چک کردن آن است.

در خط بعد، داخل p مقدار n یعنی ۱ ریخته می‌شود و هنگام چاپ، چون کاراکتر است آن کاراکتری که کد اسکی آن ۱ است چاپ خواهد شد.

بهتر است که این برنامه را خودتان بنویسید و خروجی‌ها را چک کنید. دقت کنید که بعضی از کاراکترها قابل‌دیدن نیستند مثلاً tab، space و enter این‌ها همگی هنگام چاپ به یک شکل دیده می‌شوند.

فصل سوم
مقدمات نوشتن برنامه

۳-۱ نرم‌افزار مناسب و اجرای برنامه

حال که با تعریف متغیرها و گرفتن حافظه آشنا شدیم، می‌خواهیم اولین برنامه‌ی خود را بنویسیم. قبل از آن لازم است تا ابتدا طریقه‌ی نوشتن برنامه در کامپیوتر و اجرای آن را کمی توضیح دهیم.

هر برنامه C++ شامل یک فایل با پسوند cpp . است که داخل آن، متن برنامه نوشته شده است. در واقع، در هر فایل متنی که برنامه نوشته شود و با پسوند cpp . ذخیره شود، این برنامه قابل‌قبول است؛ اما هر محیطی به منظور خاصی نوشته شده است، مثلاً، نرم‌افزار word، برای نوشتن و ویرایش متن طراحی‌شده و برای نوشتن برنامه‌ی C++ طراحی نشده و کار را دشوار خواهد کرد. از آن طرف، نرم‌افزارهایی وجود دارد که مخصوص برنامه‌نویسی هستند و کار را برای برنامه‌نویس آسان‌تر خواهند کرد. در ویندوز، نرم‌افزار dev یکی از این مثال‌ها است، هم‌چنین، eclipse، gvim، xcode در mac و اِدیتورهای فراوانی وجود دارند و قابل‌استفاده هستند. دقت کنید که بیشتر انتخاب اِدیتور سلیقه‌ای و بر اساس عادت شخص است.

حال که یک اِدیتور مناسب انتخاب کردیم، می‌توانیم داخل آن برنامه بنویسیم و با نام مورد نظر مثلاً havij . cpp آن را ذخیره کنیم.

می‌دانیم که این برنامه به زبان ما نوشته‌شده است و همچنان کامپیوتر آن را نخواهد فهمید؛ بنابراین، باید به روشی این برنامه را به زبان ماشین تبدیل کنیم (که صفر و یک است). این کار وظیفه برنامه‌ای به نام compiler است، یا به عبارتی برنامه ما باید compile شود؛ و پس از کامپایل شدن، یک نسخه از آن به زبان ماشین تولید می‌شود و پسوند آن اجرایی است برای مثال در ویندوز exe خواهد بود. حال با اجرا کردن آن فایل اجرایی برنامه اجرا خواهد شد. هم‌چنین، هنگام کامپایل کردن، کامپایلر ممکن است ایرادهایی از برنامه ما بگیرد و قابلیت ایجاد فایل اجرایی نباشد که در آن صورت باید ابتدا ایرادهایی را که داشتیم برطرف کنیم.

```
g++ havij.cpp -o golabi
```

با اجرای این خط در محیط terminal یا همان cmd، فایل havij.cpp کامپایل شده و گلابی به‌عنوان فایل اجرایی به وجود می‌آید. حال دستور زیر فایل را اجرا می‌کند:

```
./golabi
```

بعضی از اِدیتورها، داخل خودشان کامپایلر دارند و تمامی این کارها با یک دکمه اتفاق می‌افتد. در ویندوز، dev-cpp این خاصیت را دارد؛ و ما در این کتاب برای راحتی با اِدیتور dev-cpp کار می‌کنیم.

کسانی که در سیستم‌عامل دیگری برنامه‌نویسی می‌کنند یا به کار کردن با اِدیتور دیگری علاقه‌مند هستند، می‌توانند با بررسی در اینترنت کار کردن با آن‌را یاد بگیرند.

در محیط dev، برنامه نوشته می‌شود، در منوی بالا، دکمه‌ی compile وجود دارد و پس از کامپایل، دکمه‌ی run موجود است که برنامه را اجرا می‌کند. برای آشنایی بیشتر آن را نصب‌کرده و با آن کار کنید و هم‌چنین، در وب‌سایت آن می‌توانید توضیحات تکمیلی را مشاهده فرمایید.

۲-۳ مقدمات نوشتن برنامه

حال نوبت به نوشتن برنامه می‌رسد. هر برنامه C++ خط‌هایی دارد که همیشه ثابت هستند و نوشته می‌شوند. در ابتدا کتابخانه‌هایی که لازم داریم را می‌نویسیم که معمولاً یکی از آن‌ها همیشه ضروری خواهد بود. سپس یک خط همیشه ثابت داریم که std را اضافه می‌کند. پس از آن بلوک main را می‌نویسیم. در حقیقت هر برنامه C++ باید یک main داشته باشد و هنگام اجرای برنامه، همیشه داخل main شروع به اجرا شدن می‌شود.

۳-۳ بلوک

در بعضی قسمت‌ها، از جمله خود main، در شرط‌ها و هم‌چنین حلقه‌ها، پس از نوشتن یک دستور (مثلاً دستور شرط که if است)، پس از آن { باز می‌شود و با } انتهای آن مشخص می‌شود، بنابراین، تمام دستورهای داخل {} مربوط به آن شرط هستند (در مثال‌ها بهتر متوجه خواهید شد.) به هر کدام از این‌ها یک بلوک می‌گویند. برای مثال خود main هم که در پایین‌تر خواهیم دید، با { شروع می‌شود، دستورات داخل آن نوشته می‌شوند و سپس با } پایان می‌یابد و کل main یک بلوک حساب می‌شود.

نکات زیر در برنامه‌ها دقت شود:

- ✓ در C++ فاصله بیشتر از یکی، بی‌معنی بوده و برای کامپیوتر بی‌تأثیر است؛ بنابراین، گذاشتن چند فاصله هیچ مشکلی در برنامه ایجاد نمی‌کند. هم‌چنین همین موضوع برای new line یا همان enter هم برقرار است.
- ✓ بین کلمات کلیدی باید یک فاصله زده شود تا کامپیوتر دستورهای مجزا را تشخیص دهد.
- ✓ هر دستور باید با semicolon (;) خاتمه پیدا کند. در حقیقت کامپیوتر جلو می‌رود تا به ; برسد و بعد متوجه می‌شود که یک دستور باید اجرا شود.

با وجود این‌که در نوشتن برنامه‌ها، آزادی زیادی وجود دارد و باسلیقه‌های مختلف می‌توان نوشت، استانداردهایی وجود دارد که آن‌ها را رعایت کنیم. در غیر این صورت، وقتی برنامه از حدی بیشتر و پیچیده‌تر شود، به‌سختی قابل‌نوشتن خواهد بود.

۱- هر دستور در یک خط نوشته شود. درست است که از enter ها چشم‌پوشی می‌شود و کامپیوتر وجود آن‌ها را در نظر نمی‌گیرد و می‌توان کل برنامه را (۲۰۰ دستور برای مثال) در یک خط نوشت، اما به‌شدت کار را دشوار خواهد کرد؛ بنابراین هر دستور در یک خط نوشته شود.

۲- از space زدن اضافه پرهیز شود و فقط بین کلمات معنی‌دار یک space زده شود.

۳- داخل هر بلوک، به فاصله یک tab از اول بلوک، جلوتر نوشته شود.

حال برنامه ساده زیر را در نظر بگیرید که تقریباً همه برنامه‌های C++ آن‌را خواهند داشت:

```
#include <iostream>
using namespace std;
int main(){
     return 0;
}
```

همان‌طور که مشاهده می‌کنید، کتاب‌خانه‌ها با اسم‌های مختلف (در این مثال iostream) به شکل بالا اضافه می‌شوند. بلوک main حتماً باید تعریف شود. داخل {} دستورات نوشته می‌شود؛ و خط آخر main، return 0 را باید داشته باشیم. دقت کنید، همان‌طور که در بالا گفتیم، دستورات داخل بلوک main با یک tab فاصله نوشته‌شده‌اند.

دقت کنید که شروع بلوک یعنی همان‌جایی که { خواهد داشت، دیگر ؛ نداریم زیرا هنوز دستور تمام نشده است. هم‌چنین در انتهای بلوک‌ها هم یعنی جایی که با } بسته می‌شود، بازهم؛ نخواهیم داشت.

برای مثال:

```
include <iostream>
using namespace std;
int main() {
      int a;
      int b;
      int c;
      a = 8;
      b = 11;
      c = a + b;
      return 0;
}
```

برنامه کوچک بالا، یک main دارد، مقداری دستور شامل تعریف متغیر و هم‌چنین تعدادی عملیات دارد و سپس برنامه تمام می‌شود (این برنامه فقط جنبه آموزشی دارد و با اجرای آن چیزی مشاهده نمی‌شود، زیرا دستوری برای نمایش دادن استفاده‌نشده است).

بهتر است این چند دستوری که معرفی شد و برای همه برنامه‌ها ضروری است را در ابتدا حفظ کنید تا به‌مرور زمان، با ادامه درس، دلیل آن‌ها توضیح داده شود.

دقت کنید که گاهی کسانی که از dev استفاده می‌کنند ممکن است با مشکلی مواجه شود که هنگام اجرای برنامه، برنامه به‌سرعت اجراشده و صفحه بسته می‌شود و امکان مشاهده وجود نداشته باشد. در این صورت قبل از return 0 از دستور زیر استفاده کنید:

```
system ("pause");
```

با حل سؤالات از فصل بعد مفاهیم را بهتر یاد خواهید گرفت. برای این فصل، برنامه dev را دانلود کرده و نصب کنید و دو برنامه کوچک بالا را نوشته و اجرا کنید.

برای mac و linux هم داخل اِدیتور مورد نظر برنامه‌های بالا را بنویسید و سپس از terminal آن‌را همان‌طور که توضیح داده شد اجرا کنید.