

Northern University, Nowshera

Programming Fundamentals (CS)

Muhammad Athar

Email:athar@northern.edu.pk

WahatsApp: 0333-5077664

(Week 3) Lecture 5 and 6

Objectives: Learning objectives of this lecture are

- **Unary operator**
- **Char data type**
- **Integer data type**



Northern University, Nowshera

Programming Fundamentals (CS)

Muhammad Athar

Email:athar@northern.edu.pk

WahatsApp: 0333-5077664

Programming Fundamentals (CS)

Lecture # 5 Unary Operators and Char Data type

In this lecture we shall be discussing two different concepts one related with unary operator and other char data type.

Let's first look at two types of unary operators ++ (increment) and -- (decrement). ++ adds 1 in a memory and -- subtracts 1 from a memory. Unary operators requires only one operand. Both Increment and decrement are divided into further two categories prefix and postfix. When operator is used before some variable it is called prefix and when used after some variable it is called postfix. Further detail will be explained in programs given below.

Program 1:

```
#include "iostream"
using namespace std;
void main() {
```

```
    int number = 10;
```

```
    number++;
```

```
}
```

Will declare a memory with name number and assign 10 to it.

number

10

10 11

We can say number++ is equal to number = number +1;

Program 2:

```
#include "iostream"
using namespace std;
void main() {
```

```
    int number = 10;
```

```
    ++number;
```

```
}
```

Will declare a memory with name number and assign 10 to it.

number

10

10 11

We can say ++number is equal to number = number +1;

→ ++ number Prefix Increment (Pre Increment) and number++ → Postfix Increment (Post Increment). So far from the above explain it seems that there is no difference between pre and post increment/decrement. Let us dig deep into it and find difference between pre and post.

Pre Increment	Post Increment
<pre>#include "iostream" using namespace std; void main() { int n = 10; int y; y = ++n; }</pre> In this case first value of n is incremented by 1 and then assigned to y.	<pre>#include "iostream" using namespace std; void main() { int n = 10; int y; y = n++; }</pre> In this case first value of n is assigned to n and then incremented

Consider the program given.

Program	Memory
<pre>1 #include "iostream" 2 using namespace std; 3 void main() 4 { 5 int n = 10; 6 int y = n++ * ++n; 7 }</pre>	At line number 5 a memory block is created with name n and inside that memory 10 is assigned. Statement at line 6 is break down into four sub operations Operation 1 (pre increment) $n = n + 1$ (pre increment having highest priority) Memory looks like Operation 2 (Multiplication) $n * n$ (multiplication is prior to post increment) value inside memory $n = 11$ so $11 * 11 \rightarrow 121$

Northern University, Nowshera

Programming Fundamentals (CS)

Muhammad Athar

Email:athar@northern.edu.pk

WahatsApp: 0333-5077664

Programming Fundamentals (CS-323)

	<u>Operation 3 (Assignment to y)</u> y y= result of operation 2 i.e 121 121 <u>Operation 4 (Post increment)</u> n=n+1 (post increment having least priority) Memory n looks like n 10 11 12 <u>Final Value of memories:</u> n → 12 and y → 121
--	--

Practice Consider 2 programs and try to find out output

<pre>#include "iostream" using namespace std; void main() { int n = 10; int y = n++ + ++n * ++n - n-- - --n; cout << n << endl; cout << y << endl; }</pre>	<pre>#include "iostream" using namespace std; void main() { int n = 10; cout << "Value of n = " << n << endl; cout << "Post Increment on n = " << n++ << endl; cout << "After Post Increment " << n << endl; cout << "Pre increment on n " << ++n << endl; cout << "After Pre Increment " << n << endl; cout << "Post Decrement " << n-- << endl; cout << "After Post Decrement " << n << endl; cout << "Pre Decrement " << --n << endl; cout << "After Pre Decrement " << n << endl; }</pre>
Output	Output

Northern University, Nowshera

Programming Fundamentals (CS)

Muhammad Athar

Email:athar@northern.edu.pk

WahatsApp: 0333-5077664

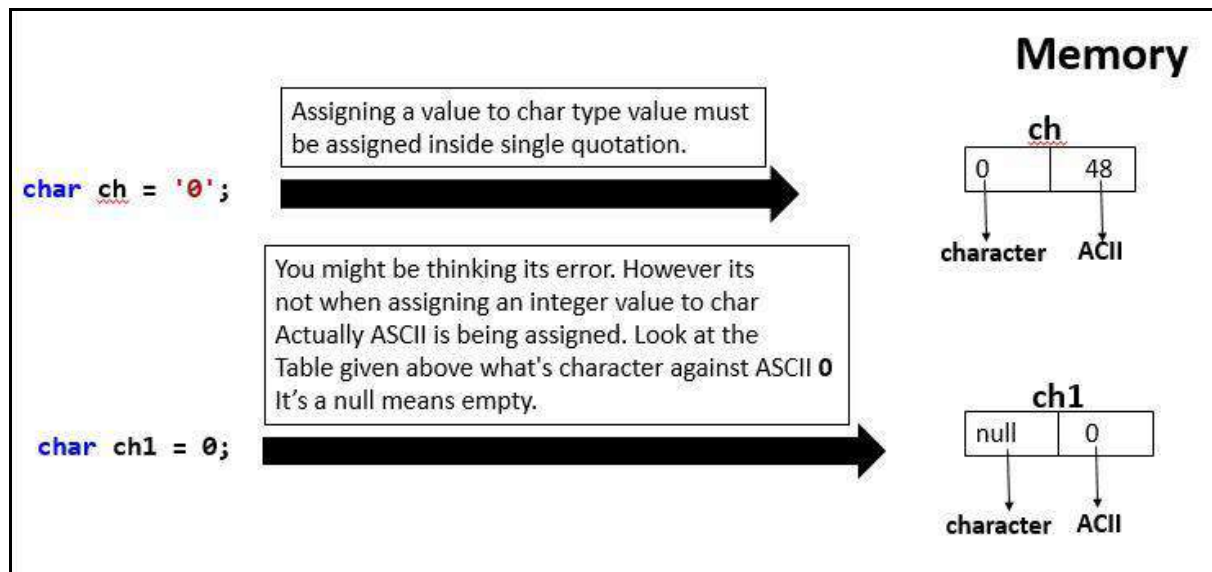
Programming Fundamentals (CS)

Now let us talk about char data type. Before moving onto a program we will first be discussing ASCII table. Every char is assigned some unique integer number called ASCII code. Following is the list of characters and their ASCII values.

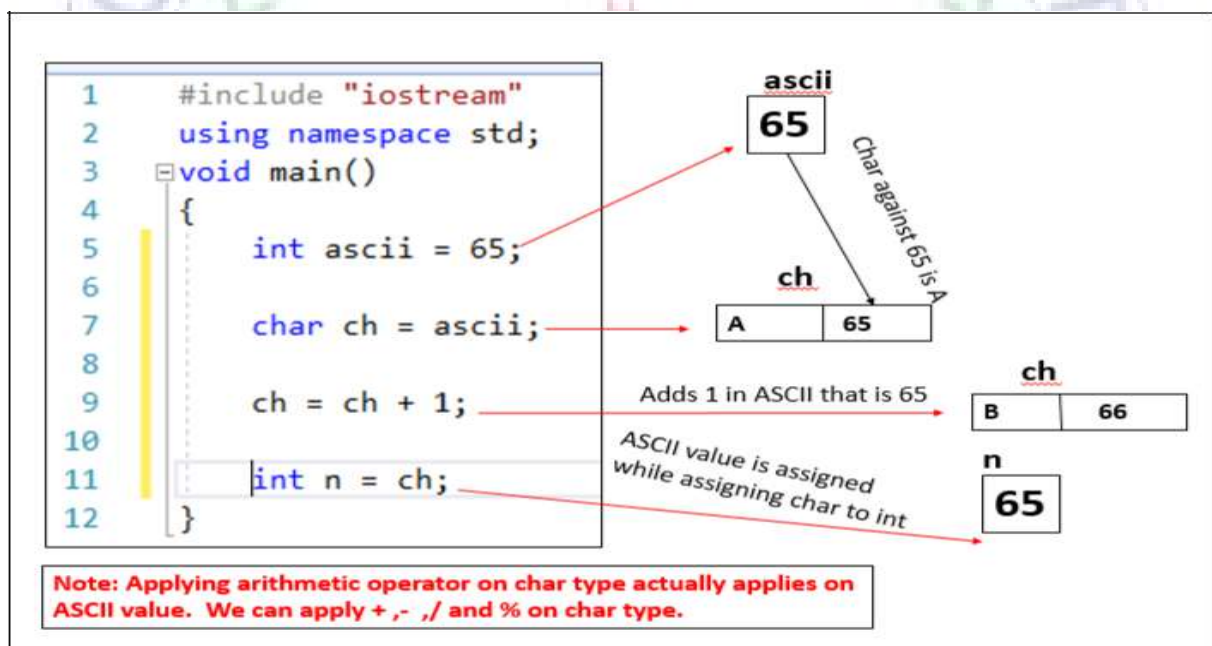
ASCII(American Standard Code for Information Interchange)

Dec	Char	Dec	Char	Dec	Char	Dec	Char
0	NUL (null)	32	SPACE	64	@	96	`
1	SOH (start of heading)	33	!	65	A	97	a
2	STX (start of text)	34	"	66	B	98	b
3	ETX (end of text)	35	#	67	C	99	c
4	EOT (end of transmission)	36	\$	68	D	100	d
5	ENQ (enquiry)	37	%	69	E	101	e
6	ACK (acknowledge)	38	&	70	F	102	f
7	BEL (bell)	39	'	71	G	103	g
8	BS (backspace)	40	(	72	H	104	h
9	TAB (horizontal tab)	41	)	73	I	105	i
10	LF (NL line feed, new line)	42	*	74	J	106	j
11	VT (vertical tab)	43	+	75	K	107	k
12	FF (NP form feed, new page)	44	,	76	L	108	l
13	CR (carriage return)	45	-	77	M	109	m
14	SO (shift out)	46	.	78	N	110	n
15	SI (shift in)	47	/	79	O	111	o
16	DLE (data link escape)	48	0	80	P	112	p
17	DC1 (device control 1)	49	1	81	Q	113	q
18	DC2 (device control 2)	50	2	82	R	114	r
19	DC3 (device control 3)	51	3	83	S	115	s
20	DC4 (device control 4)	52	4	84	T	116	t
21	NAK (negative acknowledge)	53	5	85	U	117	u
22	SYN (synchronous idle)	54	6	86	V	118	v
23	ETB (end of trans. block)	55	7	87	W	119	w
24	CAN (cancel)	56	8	88	X	120	x
25	EM (end of medium)	57	9	89	Y	121	y
26	SUB (substitute)	58	:	90	Z	122	z
27	ESC (escape)	59	;	91	[	123	{
28	FS (file separator)	60	<	92	\	124	
29	GS (group separator)	61	=	93	]	125	}
30	RS (record separator)	62	>	94	^	126	~
31	US (unit separator)	63	?	95	_	127	DEL

In the figure given above some character and their codes are highlighted. Consider following statements

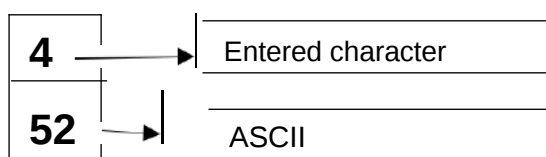


Consider following program for understanding char in more detail.



char ch;

cin>>ch; // assume that user enters 4 than this 4 is consider as character and in above table you can see ASCII value of 4. Memory will look as.



Lecture # 6 Integer and Char

Objective of this lecture is to make you more familiar with integer and char type. For detailed understanding let us look at the scenario given below and try to code for it.

Program: A bowler bowls an over of 6 balls without taking any wicket. You have to input number of runs scored on each bowl. Calculate and display total runs scored. *Assuming that values entered will either be 0,1,2,3,4 and 6.*

We shall be dealing with four different version of programs for above statement.

Version 1

Steps	Code
1. Declare 6 integer type memories for storing runs scored.	<pre>#include "iostream" using namespace std; void main() { int b1_score, b2_score, b3_sore, b4_score, b5_score, b6_score;</pre>
2. Input in above memories.	
3. Declare another memory for storing total runs Add values of all 6 memories	<pre>cout << "Enter score on ball 1,2,3,4,5 and 6 "; cin >> b1_score >> b2_score >> b3_sore >> b4_score >> b5_score >> b6_score;</pre>
4. declared in part 1 and store in memory declared in part 3.	<pre>int total; total = b1_score + b2_score + b3_sore + b4_score + b5_score + b6_score;</pre>
5. Display Memory declared in part 3.	<pre>cout << "Total Score " << total << endl; }</pre>

How many bytes of memory will be consumed? We declared 7 integer type memories. If 1 memory contains 4byte than total memory consumed is i.e $7*4= 28\text{bytes}$. Its size can be reduced to half by using short type. Can above program be converted into some other way so that memory size get reduced further?

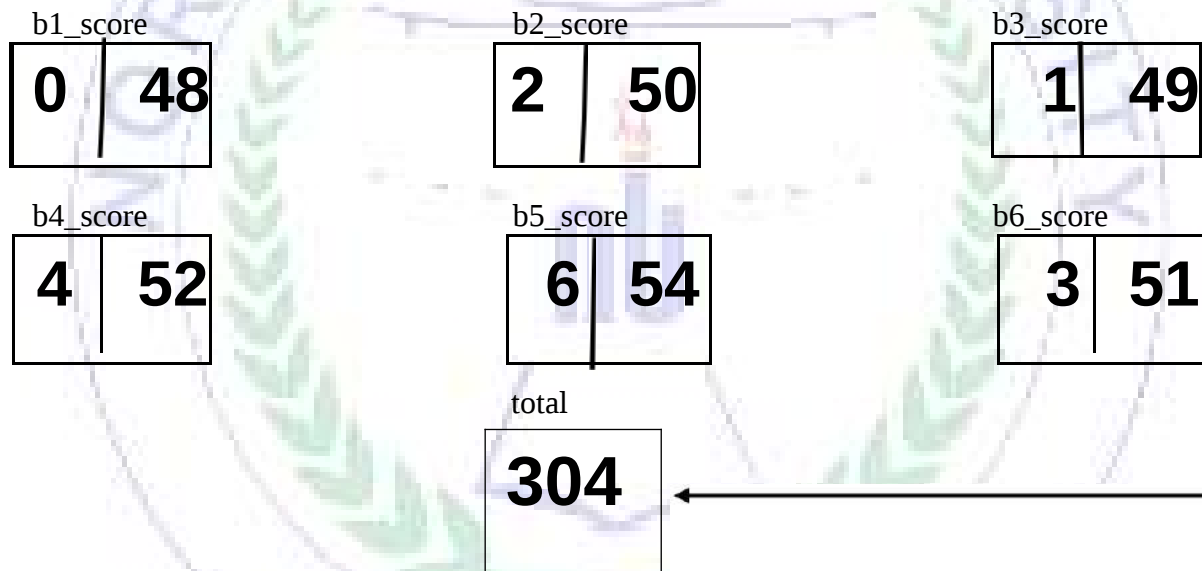
If we think of maximum runs scored on ball is 6.

So why to use int if a char is capable of storing 6.

And Maximum runs can be scored in over is 36(which cannot be stored in char so for total runs we should use int/short (short preferred))

```
1  #include "iostream"
2  using namespace std;
3  void main()
4  {
5      char b1_score, b2_score, b3_sore, b4_score, b5_score, b6_score;
6
7      cout << "Enter score on ball 1,2,3,4,5 and 6 ";
8      cin >> b1_score;
9      cin >> b2_score;
10     cin >> b3_sore;
11     cin >> b4_score;
12     cin >> b5_score;
13     cin>> b6_score;
14
15     int total;
16     total = b1_score + b2_score + b3_sore + b4_score + b5_score + b6_score;
17
18     cout << "Total Score " << total << endl;
19 }
```

Memory

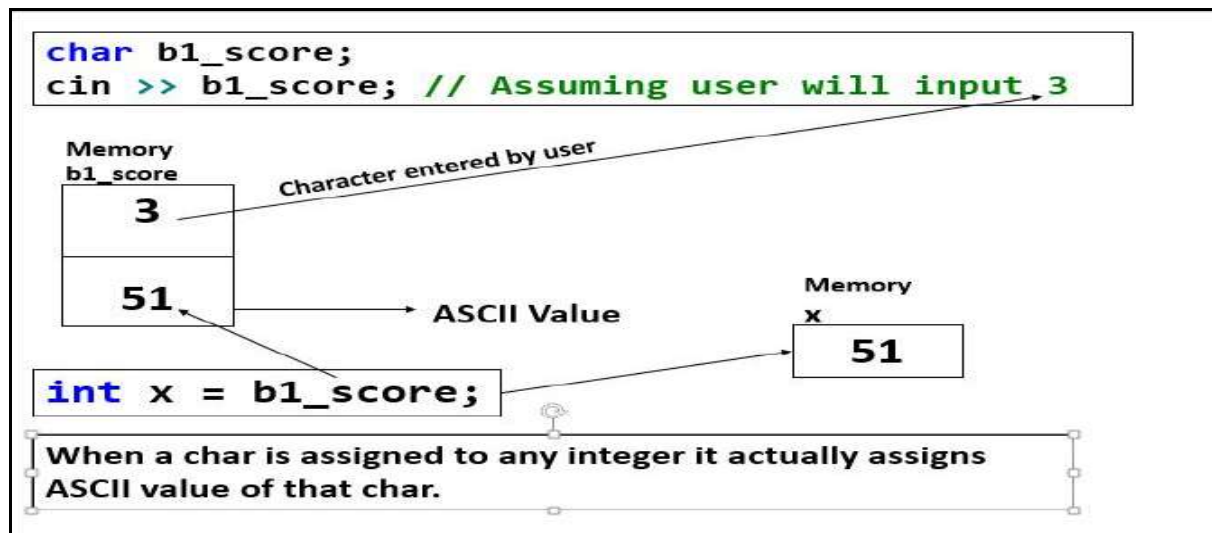


As we discussed in previous lecture that whenever an arithmetic operation is performed it is actually applied on ASCII. In above char memories single digits are char and along with them are ASCII values

`total = b1_score + b2_score + b3_sore + b4_score + b5_score + b6_score;`

48 + 50 + 49 + 52 + 54 + 51 → 304

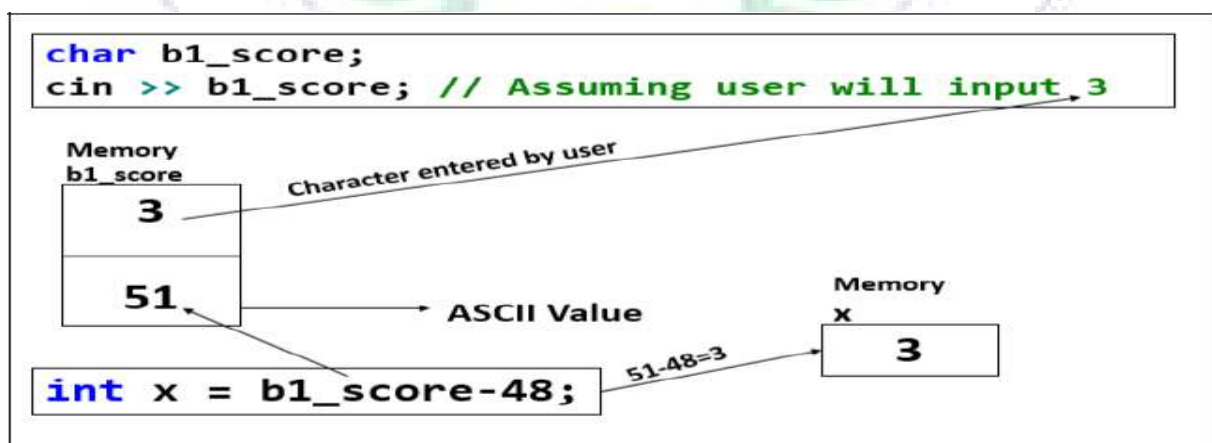
Answer is wrong because maximum runs in an over can be 36 this is because in case of char + operator adds ascii values. Now consider logic given below and will change above statement accordingly.



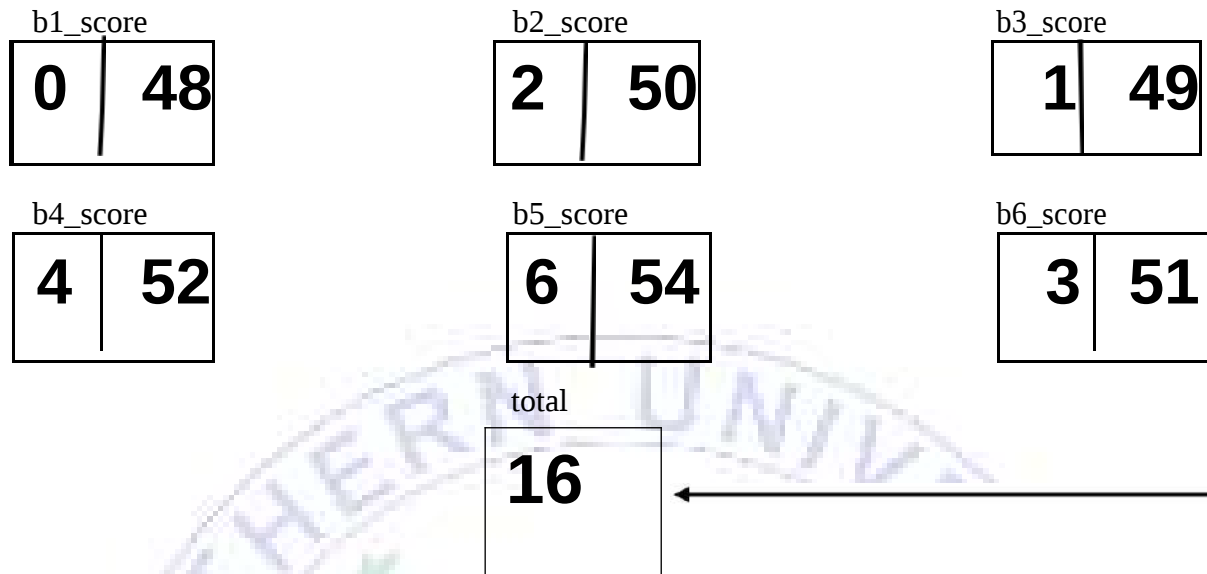
What if we want to store actual character in above case? Now consider table given below.

Char	ASCII	Subtract 48 from ASCII
0	48	48 - 48 = 0
1	49	49 - 48 = 1
2	50	50 - 48 = 2
3	51	51 - 48 = 3
4	52	52 - 48 = 4
5	53	53 - 48 = 5
6	54	54 - 48 = 6
7	55	55 - 48 = 7
8	56	56 - 48 = 8
9	57	57 - 48 = 9

From above table it is obvious if char entered by use is between 0-9 we can subtract 48 to get original char.



Memory



As we discussed in previous lecture that whenever an arithmetic operation is performed it is actually applied on ASCII. In above char memories single digits are char and along with them are ASCII values

`total = b1_score + b2_score + b3_sore + b4_score + b5_score + b6_score;`

$$(48-48) + (50-48) + (49-48) + (52-48) + (54-48) + (51-48) \rightarrow 16$$

Now total variable will have correct value.

Version 3:

Why to declare so many variables we can do it by declaring only two integer variables.

Version 3

```
1  #include "iostream"
2  using namespace std;
3  void main()
4  {
5      int total = 0, score;
6      cout << "Enter runs scored on ball 1,2,3,4,5 and 6 ";
7      cin >> score;
8      total = total + score;
9
10     cin >> score;
11     total = total + score;
12
13     cin >> score;
14     total = total + score;
15
16     cin >> score;
17     total = total + score;
18
19     cin >> score;
20     total = total + score;
21
22     cin >> score;
23     total = total + score;
24
25     cout << total;
26 }
```

Memory values

Line #	Value of total	Value of score
5	total is assigned 0	Has garbage value
7	0	User enter 2
8	2 (0+2)	2
10	2	User enters 3
11	5 (2+3)	3
13	5	User enter 0
14	5 (5+0)	0
16	5	User enter 6
17	11 (5+6)	6
19	11	User enter 2
20	13 (11+2)	2
21	13	User enter 4
22	17 (13+4)	4

Final value of total i.e 17 will be displayed on screen.

Same program can be done using char type also.

```
1  #include "iostream"
2  using namespace std;
3  void main()
4  {
5      int total = 0;
6      char score;
7      cout << "Enter runs scored on ball 1,2,3,4,5 and 6 ";
8      cin >> score;
9      total = total + (score-48); // subtracting 48 to get original character
10
11     cin >> score;
12     total = total + (score-48);
13
14     cin >> score;
15     total = total + (score-48);
16
17     cin >> score;
18     total = total + (score-48);
19
20     cin >> score;
21     total = total + (score-48);
22
23     cin >> score;
24     total = total + (score-48);
25
26     cout << total;
27 }
```

