

C++ ARRAYS

NUMBER CONVERSIONS

Problem Solving with Computers-I

C++

```
#include <iostream>
using namespace std;

int main()
{
    cout << "Hola, Facebook!";
    return 0;
}
```

GitHub

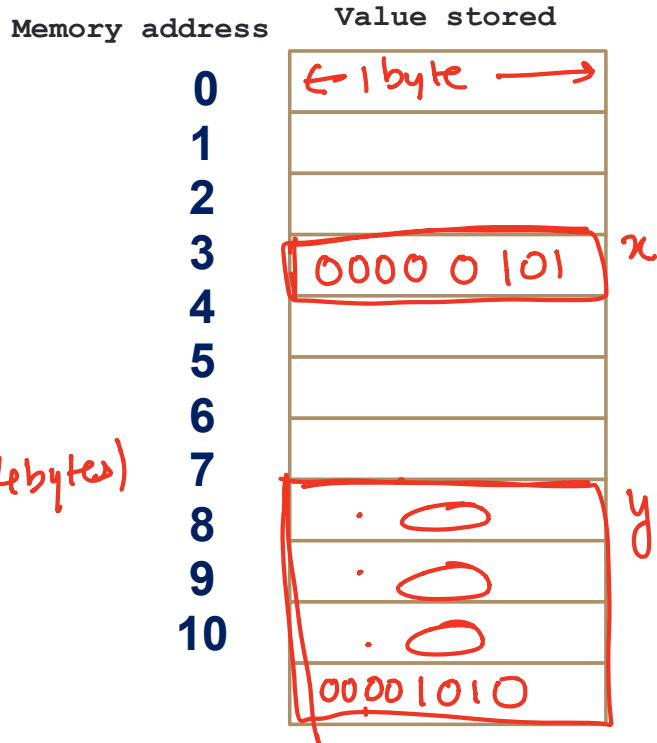


General model of memory

- Sequence of adjacent cells
- Each cell has 1-byte stored in it
- Each cell has an address (memory location)

char x = 5; (char is 1 byte)

int y = 10; (int is usually 4 bytes)



Storing sequences in programs

Write a program to take a sequence of midterm scores (out of 100) and compute the average of the midterm

C++ Arrays

A C++ array is a **list of elements** that share the same name, have the same data type and are located adjacent to each other in memory

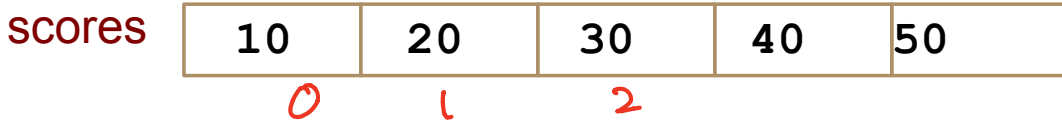
scores



Declare:

`int scores [5];`
↑
type of each element
↑
size of the array

What is the memory location of each element?



```
int scores[]={10, 20, 30, 40, 50};
```

If the starting location of the array is 0x200, what is memory location of element at index 2?

A. 0x201

B. 0x202

C. 0x204

D. 0x208

$$0x200 + 2 * \text{sizeof(int)} \\ = 0x208$$

Exercise: Reassign each value to 60



scores[0] scores[1] scores[2]

```
int scores[]={20,10,50}; // declare and initialize
```

```
//Access each element and reassign its value to 60
```

```
for (int i=0; i<3; i++) {
```

```
    scores[i]=60;
```

```
}
```

Exercise: Increment each element by 10



scores[0] scores[1] scores[2]

```
int scores[]={20,10,50}; // declare and initialize
```

```
//Increment each element by 10
```

```
for (int i=0; i<3; i++) {
```

```
    scores[i] = scores[i]+10;
```

```
}
```

Most common array pitfall- out of bound access



scores[0] scores[1] scores[2]

```
int arr[]={20,10,50}; // declare an initialize
for(int i=0; i<=3; i++)
    scores[i] = scores[i]+10;
```

Demo: Passing arrays to functions



Tracing code involving arrays



```
int arr[]={1,2,3};  
int tmp = arr[0];  
arr[0] = arr[2];  
arr[2] = tmp;
```

Choose the resulting array after the code is executed



D. None of the above

Converting between binary and decimal

Binary to decimal: $1\ 0\ 1\ 1\ 0_2 = ?_{10}$

Decimal to binary: $34_{10} = ?_2$

Hex to binary

- Each hex digit corresponds directly to four binary digits
- Programmers love hex, why?
- Convert to binary

0x25B= ?

00	0	0000
01	1	0001
02	2	0010
03	3	0011
04	4	0100
05	5	0101
06	6	0110
07	7	0111
08	8	1000
09	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Hexadecimal to decimal

$$25B_{16} = ? \text{ Decimal}$$

Hexadecimal to decimal

- Use polynomial expansion
- $25B_{16} = 2*256 + 5*16 + 11*1 = 512 + 80 + 11$
 $= 603$
- Decimal to hex: $36_{10}=?_{16}$




Binary to hex: 1000111100

A. 8F0

B. 23C

C. None of the above

BIG IDEA: Bits can represent anything!!

Numbers	Binary Code	Colors	Binary code
0		 <i>Red</i>	
1			
2		 <i>Green</i>	
3		 <i>Blue</i>	

N bits can represent at most 2^N things

What is the minimum number of bits required to represent all the letters in the English alphabet (assume only upper case)?

- A. 3
- B. 4
- C. 5
- D. 6
- E. 26



What is the maximum positive value that can be stored in a byte?

A. 127

B. 128

C. 255

D. 256

BIG IDEA: Bits can represent anything!!

- Logical values?
 - 0 \Rightarrow False, 1 \Rightarrow True
- colors ?
- Characters?
 - 26 letters \Rightarrow 5 bits ($2^5 = 32$)
 - upper/lower case + punctuation \Rightarrow 7 bits (in 8) (“ASCII”)
 - standard code to cover all the world’s languages \Rightarrow 8,16,32 bits (“Unicode”)
 - www.unicode.com
- locations / addresses? commands?

Dec	Hex	Char	Dec	Hex	Oct	Ctrl	Dec	Hex	Oct	Chr	Dec	Hex	Oct	Chr	Dec	Hex	Oct	Chr
0	000	NUL	0	000	000		64	40	100	4050	+	96	60	140	4050			
1	001	SOH	1	001	001	!	65	41	101	4051	!	97	61	141	4051	!		
2	002	STX	2	002	002	"	66	42	102	4052	"	98	62	142	4052	"		
3	003	TXL	3	003	003	#	67	43	103	4053	#	99	63	143	4053	#		
4	004	ETX	4	004	004	\$	68	44	104	4054	\$	100	64	144	4054	\$		
5	005	FXH	5	005	005	%	69	45	105	4055	%	101	65	145	4055	%		
6	006	ACK	6	006	006	&	70	46	106	4056	&	102	66	146	4056	&		
7	007	NAK	7	007	007	'	71	47	107	4057	'	103	67	147	4057	'		
8	010	BS	8	010	010	(72	48	110	4058	(104	68	148	4058	(
9	011	TAB	9	011	011)	73	49	111	4059)	105	69	149	4059)		
10	012	LF	10	012	012	*	74	4A	112	405A	*	106	6A	150	405A	*		
11	013	VT	11	013	013	+	75	4B	113	405B	+	107	6B	151	405B	+		
12	014	FF	12	014	014	,	76	4C	114	405C	,	108	6C	152	405C	,		
13	015	CR	13	015	015	-	77	4D	115	405D	-	109	6D	153	405D	-		
14	016	SO	14	016	016	.	78	4E	116	405E	.	110	6E	154	405E	.		
15	017	SI	15	017	017	:	79	4F	117	405F	:	111	6F	155	405F	:		
16	020	DLE	16	020	020	;	80	50	120	4060	;	112	70	160	4060	;		
17	021	ESC	17	021	021	<	81	51	121	4061	<	113	71	161	4061	<		
18	022	FXC	18	022	022	=	82	52	122	4062	=	114	72	162	4062	=		
19	023	FXD	19	023	023	>	83	53	123	4063	>	115	73	163	4063	>		
20	024	FXE	20	024	024	@	84	54	124	4064	@	116	74	164	4064	@		
21	025	FXF	21	025	025	A	85	55	125	4065	A	117	75	165	4065	A		
22	026	FXG	22	026	026	B	86	56	126	4066	B	118	76	166	4066	B		
23	027	FXH	23	027	027	C	87	57	127	4067	C	119	77	167	4067	C		
24	030	FXI	24	030	030	D	88	58	130	4068	D	120	78	170	4068	D		
25	031	FXJ	25	031	031	E	89	59	131	4069	E	121	79	171	4069	E		
26	032	FXK	26	032	032	F	90	5A	132	406A	F	122	7A	172	406A	F		
27	033	FXL	27	033	033	G	91	5B	133	406B	G	123	7B	173	406B	G		
28	034	FXM	28	034	034	H	92	5C	134	406C	H	124	7C	174	406C	H		
29	035	FXN	29	035	035	I	93	5D	135	406D	I	125	7D	175	406D	I		
30	036	FXO	30	036	036	J	94	5E	136	406E	J	126	7E	176	406E	J		
31	037	FXP	31	037	037	K	95	5F	137	406F	K	127	7F	177	406F	K		

Source: www.LookupTables.com

ASCII table

- REMEMBER: N bits \Leftrightarrow at most 2^N things

Next time

- Pointers
- Mechanics of function calls – call by value and call by reference