

C++ ARRAYS

NUMBER CONVERSIONS

Problem Solving with Computers-I

C++

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

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



General model of memory

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

| Memory address | Value stored |
|----------------|--------------|
| 0 | |
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | |
| 6 | |
| 7 | |
| 8 | |
| 9 | |
| 10 | |

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

| | | | | | | | |
|----|----|----|----|----|--|--|--|
| 10 | 20 | 30 | 40 | 50 | | | |
|----|----|----|----|----|--|--|--|

Declare:

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
```

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
```

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;
```

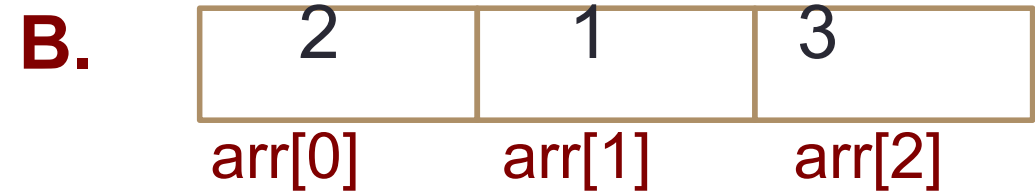
Demo: Passing arrays to functions

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

What is the memory location of each element?

scores

| | | | | |
|----|----|----|----|----|
| 10 | 20 | 30 | 40 | 50 |
|----|----|----|----|----|

```
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

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

| Dec | Hex | Oct | Char | Dec | Hex | Oct | Char | Dec | Hex | Oct | Char | | | | | | | | |
|-----|-----|-----|------------|------------------------|-----|-----|------|------|--------|-----|------|-----|------|---|-----|----|-----|------|---|
| 0 | 0 | 000 | NUL | initia | 32 | 20 | 140 | 0332 | 2space | 64 | 40 | 100 | 0354 | 1 | 96 | 60 | 140 | 0355 | |
| 1 | 1 | 001 | SOH | (start of heading) | 33 | 21 | 141 | 0333 | ! | 65 | 41 | 101 | 0355 | 2 | 97 | 61 | 141 | 0357 | 0 |
| 2 | 2 | 010 | STX | (start of text) | 34 | 22 | 142 | 0334 | " | 66 | 42 | 102 | 0356 | F | 98 | 62 | 142 | 0358 | 0 |
| 3 | 3 | 011 | ETX | (end of text) | 35 | 23 | 143 | 0335 | 3 | 67 | 43 | 103 | 0357 | C | 99 | 63 | 143 | 0359 | 0 |
| 4 | 4 | 100 | TXL | (end of transmission) | 36 | 24 | 144 | 0336 | 6 | 68 | 44 | 104 | 0358 | T | 100 | 64 | 144 | 0360 | 1 |
| 5 | 5 | 101 | VT | (vertical tab) | 37 | 25 | 145 | 0337 | 8 | 69 | 45 | 105 | 0359 | E | 101 | 65 | 145 | 0361 | 0 |
| 6 | 6 | 110 | ACK | (acknowledge) | 38 | 26 | 146 | 0338 | 7 | 70 | 46 | 106 | 0360 | 7 | 102 | 66 | 146 | 0362 | 0 |
| 7 | 7 | 111 | BS | (backspace) | 39 | 27 | 147 | 0339 | 9 | 71 | 47 | 107 | 0361 | 0 | 103 | 67 | 147 | 0363 | 0 |
| 8 | 8 | 100 | HT | (horizontal tab) | 40 | 28 | 150 | 0340 | (| 72 | 48 | 110 | 0372 | E | 104 | 68 | 150 | 0364 | 0 |
| 9 | 9 | 011 | TAB | (horizontal tab) | 41 | 29 | 151 | 0341 | 1 | 73 | 49 | 111 | 0373 | 1 | 105 | 69 | 151 | 0365 | 1 |
| 10 | A | 110 | LF | (line feed, new line) | 42 | 2A | 152 | 0342 | 2 | 74 | 4A | 112 | 0374 | 2 | 106 | 70 | 152 | 0366 | 2 |
| 11 | B | 111 | VT | (vertical tab) | 43 | 2B | 153 | 0343 | + | 75 | 4B | 113 | 0375 | E | 107 | 71 | 153 | 0367 | 2 |
| 12 | C | 010 | FF | (form feed, new page) | 44 | 2C | 154 | 0344 | . | 76 | 4C | 114 | 0376 | - | 108 | 72 | 154 | 0368 | 2 |
| 13 | D | 011 | CR | (carriage return) | 45 | 2D | 155 | 0345 | 5 | 77 | 4D | 115 | 0377 | Y | 109 | 73 | 155 | 0369 | 2 |
| 14 | E | 100 | SO | (shift out) | 46 | 2E | 156 | 0346 | 6 | 78 | 4E | 116 | 0378 | X | 110 | 74 | 156 | 0370 | 2 |
| 15 | F | 101 | SI | (shift in) | 47 | 2F | 157 | 0347 | 7 | 79 | 4F | 117 | 0379 | 0 | 111 | 75 | 157 | 0371 | 2 |
| 16 | 10 | 000 | DLE | (data link escape) | 48 | 30 | 160 | 0348 | 8 | 80 | 50 | 120 | 0380 | 8 | 112 | 76 | 160 | 0372 | 2 |
| 17 | 11 | 001 | DC1 | (device control 1) | 49 | 31 | 161 | 0349 | 9 | 81 | 51 | 121 | 0381 | 0 | 113 | 77 | 161 | 0373 | 2 |
| 18 | 12 | 010 | DC2 | (device control 2) | 50 | 32 | 162 | 0350 | 2 | 82 | 52 | 122 | 0382 | E | 114 | 78 | 162 | 0374 | 2 |
| 19 | 13 | 011 | DC3 | (device control 3) | 51 | 33 | 163 | 0351 | 3 | 83 | 53 | 123 | 0383 | 0 | 115 | 79 | 163 | 0375 | 2 |
| 20 | 14 | 100 | DC4 | (device control 4) | 52 | 34 | 164 | 0352 | 4 | 84 | 54 | 124 | 0384 | T | 116 | 80 | 164 | 0376 | 2 |
| 21 | 15 | 101 | NAK | (negative acknowledge) | 53 | 35 | 165 | 0353 | 3 | 85 | 55 | 125 | 0385 | U | 117 | 81 | 165 | 0377 | 2 |
| 22 | 16 | 000 | SYN | (synchronous idle) | 54 | 36 | 166 | 0354 | 4 | 86 | 56 | 126 | 0386 | 0 | 118 | 82 | 166 | 0378 | 2 |
| 23 | 17 | 001 | END | (end of trans. block) | 55 | 37 | 167 | 0355 | 5 | 87 | 57 | 127 | 0387 | Y | 119 | 83 | 167 | 0379 | 2 |
| 24 | 18 | 010 | FRM | (remote) | 56 | 38 | 170 | 0356 | 6 | 88 | 58 | 130 | 0388 | X | 120 | 84 | 170 | 0380 | 2 |
| 25 | 19 | 011 | EX | (end of medium) | 57 | 39 | 171 | 0357 | 7 | 89 | 59 | 131 | 0389 | 8 | 121 | 85 | 171 | 0381 | 2 |
| 26 | 1A | 100 | SUB | (substitute) | 58 | 3A | 172 | 0358 | 8 | 90 | 5A | 132 | 0390 | 7 | 122 | 86 | 172 | 0382 | 2 |
| 27 | 1B | 101 | ESC | (escape) | 59 | 3B | 173 | 0359 | 9 | 91 | 5B | 133 | 0391 | 0 | 123 | 87 | 173 | 0383 | 2 |
| 28 | 1C | 000 | FS | (file separator) | 50 | 3C | 174 | 0360 | 0 | 92 | 5C | 134 | 0392 | 2 | 124 | 88 | 174 | 0384 | 2 |
| 29 | 1D | 001 | GS | (group separator) | 51 | 3D | 175 | 0361 | 1 | 93 | 5D | 135 | 0393 | 3 | 125 | 89 | 175 | 0385 | 2 |
| 30 | 1E | 010 | RS | (record separator) | 52 | 3E | 176 | 0362 | 2 | 94 | 5E | 136 | 0394 | 4 | 126 | 90 | 176 | 0386 | 2 |
| 31 | 1F | 011 | US | (unit separator) | 53 | 3F | 177 | 0363 | 3 | 95 | 5F | 137 | 0395 | 5 | 127 | 91 | 177 | 0387 | 2 |

Source: www.LookupTables.com

ASCII table

- **REMEMBER:** N bits ⇔ at most 2^N things

Next time

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