Basics3 – Pointers

Due Date

- See Piazza for any changes to due date and time
    - Friday by midnight
    - Grading the next day Saturday Morning
- Submit program to perforce in your student directory
    - Sub directory called:
        - /Basics3/...
    - Fill out your ***Basics3 Submission Report.pdf***
        - Place it in the same directory as your solution
        - Enter the final Changelist number of your submission
        - Enter the number of test passed
        - Write up a quick discussion in the report
            - What you learned from this Basics

Goals

- C++ pointers
    - Saving the world one dereference at a time.
    - Increasing C++ knowledge and understanding

Assignments

- General:
    - Add code to the body of the functions:
        - Students_PointerWalk()
        - Students_Casting()
    - Run the Unit Tests to verify progress / success
        - 5/5 is the best for this program

- Students_PointerWalk()
    - Code up the pointer test from class (See Below)
        - Please code and step through each of these steps
        - Verify with break points and memory windows
        - This is for your benefit.
            - Please do so...

- Students_Casting()
    - Understand the 3 structures, Cat, Bird, and Dog.
    - Understand how they are added arranged inside the ***petStore*** structure.
        - Pay particular attention to the padding and alignment
    - Code the questions 1-19
        - Restrict your answers to the rules/guidelines presented in code

  - You should be able to answer those questions by paper first
    - Then verify with the code.
    - Make sure you understand these questions / relationships.

- Check in the problems multiple times, at least 3 times for this Basics assignment
  - Have reasonable check-in comments

- Make sure that your program compiles and runs
  - Warning level ALL sometimes that is not possible due to MS headers…
    - There are corrections around windows headers
  - Your code should be squeaky clean.

- Submit program to perforce in your student directory
  - Sub directory called: /Basics3/…

Validation

Simple check list to make sure that everything is checked in correctly

- Did you do all run all unit tests problems?
- Do they compile and run without any errors?
- Warning level /Wall free?
- Submitted it into /Basics3 directory - without the extra files?
- Submit the submission report?
- Can you delete your local drive, regrab the Basics3 directory?
  - Is all the code there, so that it compiles?

Hints

Most assignments will have hints in a section like this.

- This is pretty easy Basic assignment
  - It is mainly here to help you single step through your code and understand pointers layouts and access commands.
  - The casting section, allows you to access parts of a complicated structure with casting.
    - Note the data is the same, but the way you access changes.
- I expect this assignment to be completed quickly for most of the students
  - Please make sure you fully understand this code without a debugger.
  - Many little lessons here for those who put in the effort.
    - Something similar in the exam
- Enjoy

Pointer Test / Keenan

```
Assume that we are working on a LITTLE endian processor
unsigned char data[];
Memory Dump ( values in Hex )

data =     0x0000: 0xEB, 0xCD, 0x22, 0x4F,
           0x0004: 0x73, 0xB5, 0xF3, 0x35,
           0x0008: 0x23, 0x24, 0x01, 0xFE,
           0x000C: 0xCD, 0xE3, 0x44, 0x85,
           0x0010: 0x66, 0x43, 0x75, 0x33,
           0x0014: 0x39, 0x5C, 0x22, 0x11,
           0x0018: 0x56, 0xA8, 0xAA, 0x13,
           0x001C: 0x64, 0x82, 0x68, 0x26,

unsigned char  *p;  // char are 8-bits wide
unsigned int   *r;  // ints are 32-bits wide
unsigned short *s;  // shorts are 16-bits wide

p = &data[0];                                  Expected output
printf(“%x\n”,  *(p+3)  );     1)__________________________
printf(“%x\n”,  *(p+5)  );     2)__________________________
p = p + 12;
printf(“%x\n”,   *(p)  );      3)__________________________
printf(“%x\n”,   p[2]  );      4)__________________________
printf(“%x\n”,   *p++  );      5)__________________________
p += 6;
printf(“%x\n”,   *--p  );      6)__________________________
printf(“%x\n”,   p[5]  );      7)__________________________
p = p + 2;
printf(“%x\n”,   *p++  );      8)__________________________
printf(“%x\n”,  *(p+3) );      9)__________________________
p = 5 + p;
printf(“%x\n”,  *(p++) );     10)__________________________
printf(“%x\n”,  *(--p) );     11)__________________________
```

```
data =     0x0000: 0xEB, 0xCD, 0x22, 0x4F,
           0x0004: 0x73, 0xB5, 0xF3, 0x35,
           0x0008: 0x23, 0x24, 0x01, 0xFE,
           0x000C: 0xCD, 0xE3, 0x44, 0x85,
           0x0010: 0x66, 0x43, 0x75, 0x33,
           0x0014: 0x39, 0x5C, 0x22, 0x11,
           0x0018: 0x56, 0xA8, 0xAA, 0x13,
           0x001C: 0x64, 0x82, 0x68, 0x26,

r = (unsigned int *)&data[0]
printf(“%x\n”,   *(r)  );        12)__________________________
printf(“%x\n”,  *(r+5) );        13)__________________________
r++;
printf(“%x\n”,   *r++  );        14)__________________________
r = r + 2;
printf(“%x\n”,   r[2]  );        15)__________________________
r = r + 1;
printf(“%x\n”,   r[0]  );        16)__________________________

s = (unsigned short *) r;
printf(“%x\n”,   s[-2] );        17)__________________________
s = s - 3;
printf(“%x\n”,   s[2]  );        18)__________________________
s += 5;
printf(“%x\n”,  *(s+3) );        19)__________________________
printf(“%x\n”,   *(s)  );        20)__________________________
p = (unsigned char *) s;
printf(“%x\n”,  *(p+3) );        21)__________________________
p += 5;
printf(“%x\n”,  p[-9] );         22)__________________________
--p;
printf(“%x\n”,   p[0] );         23)__________________________
```