



Recitation Class 08 for VG101

Date: 2012 / 11 / 13

Wang Qian

feedback & announcement

- Take your flash disk during the lab time.
- MID 02
 - The mid-exam 2 will come next week!
- LAB 06
 - Not difficult at all, since what you learnt not so much last week.
- HWK05
 - Please pay enough attention when you submit your hwk.
 - Possible HC violation again.

Pointer

- **One of the most difficult parts in this course!!!**
- Practice makes perfect.
- Be patient to understand the whys and wherefores.
- Always ask yourself the following question:
 - Where does it point to ?
 - What is the value in it?

Pointer

- Declaration
 - `data_type *name_1,*name_2,...;`
- Initialization
 - `name = &x;` (x should have the same data_type as *name)
 - `name = (data_type*)malloc(sizeof(data_type)*length)`
 - You should include the standard library.
 - Do not forget to free the memory by using `free(name)`.
 - When the length is a variable, it still works.
 - Notice that you can never do this:
 - `int arr[n];` (even in C++)
- Call the value
 - Use `*name` to call the value stored in the memory it points to.

Quiz

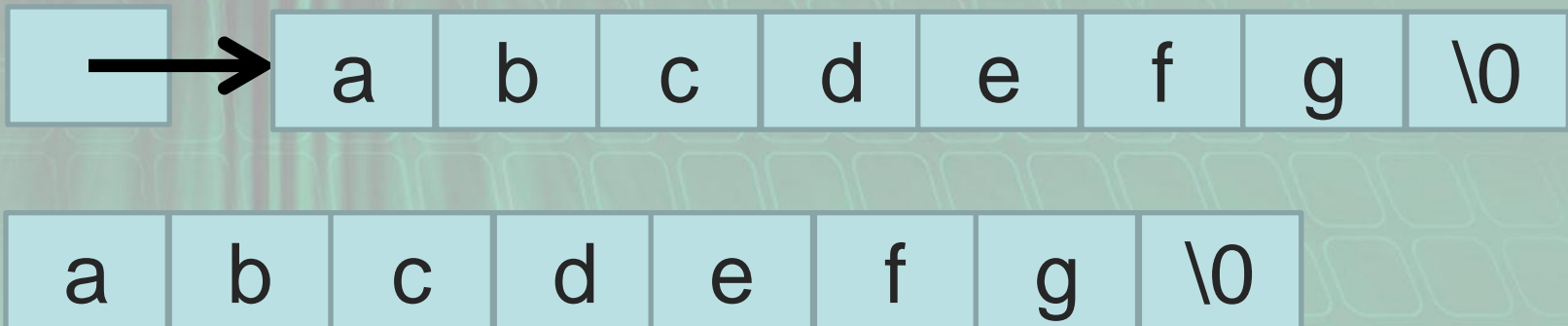
- What is the value of a after each step?
 - `int x[5] = {0,1,2,3,4},a,*pt;`
 - `pt = &x[0];`
 - `a = ++*pt;`
 - `a = *++pt;`
 - `a = *pt++;`
 - `a = (*pt)++;`
- Recall the importance of brackets.

Pointer and Array

- Close relation between pointer and array.
 - `int x[5] = {0,1,2,3,4}, *pt;`
 - `pt = x;` `// pt = &x[0];`
 - `*pt = *(x + 3);` `// *pt = x[3];`
- Recall the importance of the array boundary.
- Notice:
 - It is valid for us to use: `pt++`;
 - But illegal for use to use: `x++`;
 - We cannot modify any constant value.

Char Pointer

- Notice the difference between char array.
 - `char *stra = "abcdefg";`
 - `char stra[] = "abcdefg";`
 - `*stra = '@';`
- Under which circumstance it is valid?
- Still follow the rule that we cannot modify constant value.



Pass by Reference

- Pass by value

```
void swap(int a, int b)
{
    int t = a;
    a = b;
    b = t;
    return;
}
```

swap(x, y) ;

- Pass by reference

```
void swap(int *a, int *b)
{
    int t = *a;
    *a = *b;
    *b = t;
    return;
}
```

swap(&x, &y) ;

Quiz

- What does the following function do?
 - `void func(char *s,char *t)`
 - `{`
 - `while (*s++ = *t++);`
 - `}`
- The following format is better. Why?
 - `void func(char *s,const char *t)`
 - `{`
 - `while (*s++ = *t++);`
 - `}`

Return a Pointer

- Cannot return an array. Use the pointer instead.
- Which one is meaningful? Though both are valid... (lifespan)

```
char *func()  
{  
    char *s = "Hello, world!";  
    return s;  
}
```

```
char *func()  
{  
    char s[] = "Hello, world!";  
    return s;  
}
```

- Sometimes, it seems to work well, but trust me that the result really depends on the compiler and the state of system!

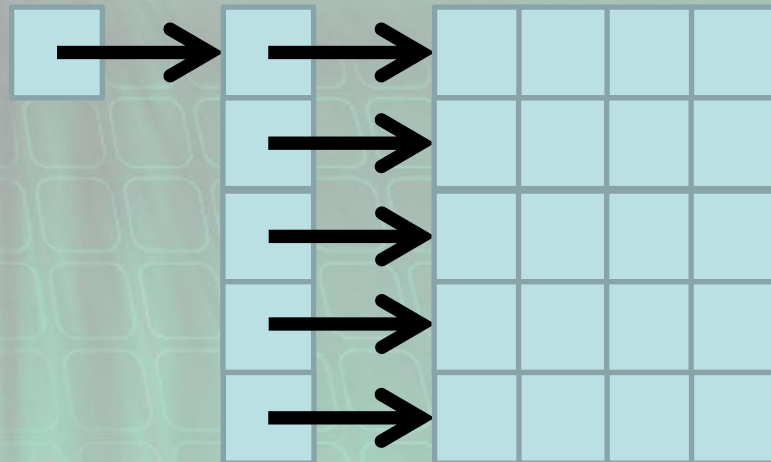
Return a Pointer

- You should return a pointer pointing to:
 - (1) a static variable
 - (2) a variable with dynamic allocated memory
 - (3) global variable
 - (4) constant variable

Pointer to Pointer

- Compare that with two-dimensional array (n=5 m=4)

```
y = (int**)malloc(n * m * sizeof(int));  
y[0] = (int*)malloc(m * sizeof(int));  
for (i = 1; i < n; i++)  
    y[i] = y[i-1] + m;
```



Structure

- Class in C++ will be similar.
- With structure, we will find the power of pointer.
 - `struct type_name {`
 - `data_type member;`
 - `};`
 - `struct type_name var_name;`
- For me, I like the following way more:
 - `typedef struct {`
 - `data_type member;`
 - `} type_name;`
 - `type_name var_name;`

Self-pointing structure

- Considering such a structure:
 - `typedef struct type_name {`
 - `struct type_name *fore,*next;`
 - `int data;`
 - `} type_name;`
- How to use this to form a loop structure?
- More data structure will be covered in Ve280.
- You can find the source code of BST I introduced during last RC, if your are interested in it and have spare time.
- I also upload the source code of merge sort. That is also not required but only for your interest.