14 Spring CS 31 TA: Zhou Ren

Questions on Project 5

Review of the Lectures Last Week

Address of a variable:

- Each variable is stored in the main memory, so it has an address.
- How to get the address? Use the reference operator, an ampersand sign (&), which can be literally translated as "address of".
- How about array? The array name itself holds an address, the address of the first element in the array.
  See Example 5.

Pointers:

- Pointers are designed to hold memory addresses. With pointer variables you can indirectly manipulate data stored in other variables.
- Pointer variables declaration: Syntax: type_name * pointer_name;
- Pointer assignment? Syntax: pointer_name = & variable_name;
  See Example 6
- Dereference: to get the value pointed by a pointer, using an asterisk (*), * pointer_name;
  The operator performed on * pointer_name is performed as if on the variable the pointer points to. See Example 7
- Use pointer as arguments of a function?? The same as pass-by-reference
  void multiplyBy2(int* val); See Example 8:
- Array and pointers, see Example 9

Example:

Example #1: How to append a character to a cstring

```
#include <iostream>
#include <cstring>

int main()
{
        // how to append a character into a cstring
        char cstr[80] = "Hello, world!";
        char c = 'K';

        int leng = strlen(cstr);
        cstr[leng] = c;
        cstr[leng + 1] = '\0';

        return 0;
}
```

Example #2: How to delete non-letter characters in a string and separate consecutive words with a single space? Assume the array of character is long enough. E.g., `"~!Hello, &&@world!%" = > "Hello world"`

```
#include <iostream>
#include <cstring>

void processString(char []);

int main()
{
	// how to append a character into a cstring
	char cstr[80] = "~!Hello, &&@world!%";
	processString(cstr);
	return 0;
}

bool isAlpha(char c)
{
	return (c > 'a' && c < 'z') || (c > 'A' && c < 'Z');
}
void processString(char str[])
{
	int leng = strlen(str);
	int i, j;
	bool prev_alpha = true;

	j = 0;
	while( !isalpha(str[j]) && j < leng )
		j++;

	i = 0;
	for(; j < leng; j++)
	{
		if( isAlpha(str[j]) )
		{
			str[i++] = str[j];
			prev_alpha = true;
		}
		else
		{
			if(prev_alpha)
				str[i++] = ' ';
			prev_alpha = false;
		}
	}

	if (str[i-1] == ' ')
		str[i-1] = '\0';
	else
		str[i] = '\0';

}
```

Example #3: Given a cstring, find the starting and ending positions of the words

----

```cpp
#include <iostream>
#include <cstring>

const int MAX_LENGTH = 80;

int parseWord(const char str[], int wordIndices[][MAX_LENGTH], int& strCharLeng);

int main()
{
	// how to append a character into a cstring
	char cstr[80] = "~!Hello, &&@world!%";
	int wordIndices[2][MAX_LENGTH];
	int charCount;
	int wordCount = parseWord(cstr, wordIndices, charCount);
	return 0;
}

bool isAlpha(char c)
{
	return (c > 'a' && c < 'z') || (c > 'A' && c < 'Z');
}
int parseWord(const char str[], int wordIndices[][MAX_LENGTH], int& strCharLeng)
{
	int leng = strlen(str);
	int wordLeng = 0;
	strCharLeng = 0;

	int idx = 0;
	for(int i=0; i<leng; i++)
	{
		if(isAlpha(str[i]))
		{
			strCharLeng ++;
			int j = i + 1;
			while ( j < leng && isAlpha(str[j]))
			{
				strCharLeng ++;
				j++;
			}

			wordIndices[0][wordLeng] = i;
			wordIndices[1][wordLeng] = j-1;
			wordLeng ++;

			i = j;
		}
	}


}
```

Example #4: given 2 words, find out whether they are a possible pair of ciphertext and plaintext? Assume all the characters are lower case. e.g.,

```
"michillinda" vs "doeiossofrq" is a valid pair because we can substitute
m->d, i->o, c->e, h->i, l->s, n->f, d->r, a->q

"meeet" vs "hello" is not a valid pair
```

```
Code not available.. How can we solve this?
```

Example #5: address and the reference operator

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

void main(void)
{
       int x = 25;
       cout << "The address of x is " << &x << endl;
       cout << "The size of x is " << sizeof(x) << " bytes\n";
       cout << "The value in x is " << x << endl;

       cout << endl << "below we try array " << endl;
       int arr[] = {1, 2,  3,  4,  5};
       cout << "The address of arr is " <<  arr << endl;
}
```

Example #6: pointer declaration and assignment

```
#include <iostream>

using namespace std;

void main(void)
{
       int x = 25;
       int *ptr;

       ptr = &x;   // Store the address of x in ptr
       cout << "The value in x is " << x << endl;
       cout << "The address of x is " << ptr << endl;
       cout << "The address of x is " << &x << endl;
}
```

Example #7: dereference of a pointer

```
#include <iostream>

void main(void)
{
	int x = 25, y = 50, z = 75;
	int *ptr;
	cout << "Here are the values of x, y, and z:\n";
	cout << x << "   " << y << "   " << z << endl;
	ptr = &x;  // Store the address of x in ptr
	*ptr *= 2; // Multiply value in x by 2
	ptr = &y;  // Store the address of y in ptr
	*ptr *= 2;  // Multiply value in y by 2
	ptr = &z;   // Store the address of z in ptr
	*ptr *= 2;  // Multiply value in z by 2
	cout << "Once again, here are the values of x, y, and z:\n";
	cout << x << "   " << y << "   " << z << endl;
}
```

Example #8: Use pointer as argument for a function

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

void multiply2ByValue(int val)
{
	val = val * 2;
}

void multiply2ByPointer(int* pval)
{
	*pval = *pval * 2;
}

void multiply2ByReference(int& val)
{
	val = val * 2;
}

int main()
{
	int val = 1;
	multiply2ByValue(val);
	cout << val << endl;

	val = 1;
	multiply2ByReference(val);
	cout << val << endl;

	val = 1;
	multiply2ByPointer(&val);
	cout << val << endl;

	return 0;
}
```

Example #9:   Pointer vs array

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

int main()
{

       int array[] = {1, 2, 3, 4, 5};

       cout << "here we test the array" << endl;
       cout << array << endl;
       cout << *array << endl;

       for (int i=0; i<5; i++)
              cout << array+i << "\t";
       cout << endl;

       for (int i=0; i<5; i++)
              cout << *(array+i) << "\t";
       cout << endl;


       int* pArray = array;

       // below test the pointer

       cout << "here we test the pointer" << endl;
       cout << pArray << endl;
       cout << *pArray << endl;

       for (int i=0; i<5; i++)
              cout << pArray+i << "\t";
       cout << endl;

       for (int i=0; i<5; i++)
              cout << *(pArray+i) << "\t";
       cout << endl;

}
```