Cuong Hoang

CSC 3320 System Level Programming

Lab Assignment 9 – Post Lab

Part 1:

Output:

```
[choang7@gsuad.gsu.edu@snowball Lab9]$ cat test.txt
This is a list of courses.
CSC 1010 - COMPUTERS & APPLICATIONS
[choang7@gsuad.gsu.edu@snowball Lab9]$ ./freq test.txt
The mos frequent letter is 's'. It appeared 8 times.
[choang7@gsuad.gsu.edu@snowball Lab9]$
```

Source Code:

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    char character;
    int pos = 0, maxFreq = 0, f = 0, tf = 0;
    int equalFreq[27];
    int freq[27];

    memset(freq, 0, sizeof(freq));
    memset(freq, 0, sizeof(equalFreq));

    FILE *filePointer;

    if (argc >= 2) {
      filePointer = fopen(argv[1], "r");
    }
    else {
      filePointer = fopen("test.txt", "r");
    }

    while ((character = fgetc(filePointer)) != EOF) {
        if (character >= 'a' && character <= 'z') {
            pos = character - 'a';
            freq[pos]++;
        }
```

```
        else if (character >= 'A' && character <= 'Z') {
            pos = character - 'A';
            freq[pos]++;
        }
    }

    for (f = 0; f < 26; f++) {
        if (freq[f] > maxFreq) {
            pos = f;
            maxFreq = freq[f];
            tf = 0;
        }
        if (maxFreq == freq[f] && freq[f] > 0) {
            equalFreq[tf] = f;
            tf++;
        }
    }

    printf("The mos frequent letter is ");
    for (f = 0; f < tf; f++) {
        printf("'%c'", equalFreq[f] + 'a');
        printf(". It appeared %d times.\n", maxFreq);
    }

    fclose(filePointer);
    return 0;
}
```

Part 2:

1) Output:

```
[choang7@gsuad.gsu.edu@snowball Lab9]$ ./addressOfScalar
address of charvar = 0x7ffc653ecd0f
address of charvar - 1 = 0x7ffc653ecd0e
address of charvar + 1 = 0x7ffc653ecd10
address of intvar = 0x7ffc653ecd08
address of intvar - 1 = 0x7ffc653ecd04
address of intvar + 1 = 0x7ffc653ecd0c
[choang7@gsuad.gsu.edu@snowball Lab9]$
```

2) Source Code:

```c
#include <stdio.h>

int main() {
    // intialize a char variable, print its address and the next address
    char charvar = '\0';
    printf("address of charvar = %p\n", (void *)(&charvar));
    printf("address of charvar - 1 = %p\n", (void *)(&charvar - 1));
    printf("address of charvar + 1 = %p\n", (void *)(&charvar + 1));

    // intialize an int variable, print its address and the next address
    int intvar = 1;
    printf("address of intvar = %p\n", (void *)(&intvar));
    printf("address of intvar - 1 = %p\n", (void *)(&intvar - 1));
    printf("address of intvar + 1 = %p\n", (void *)(&intvar + 1));
}
```

3)

The address after **intvar** is incremented by 4 bytes instead of 1 byte because the variable is of the data type integer, which takes 4 bytes memory in a 32-bit or 64-bit C compiler.

Part 3:

1) Output:

```
[choang7@gsuad.gsu.edu@snowball Lab9]$ gcc addressOfArray.c  -o addressOfArray
[choang7@gsuad.gsu.edu@snowball Lab9]$ ./addressOfArray
numbers = 0x7ffe5eea49c0
numbers[0] = 0x7ffe5eea49c0
numbers[1] = 0x7ffe5eea49c4
numbers[2] = 0x7ffe5eea49c8
numbers[3] = 0x7ffe5eea49cc
numbers[4] = 0x7ffe5eea49d0
sizeof(numbers) = 20
[choang7@gsuad.gsu.edu@snowball Lab9]$
```

2) The address of the array and the address of the first element in the array **are the same**.

3) Statement to print out the length of the array using **sizeof** operator:

```
int length = sizeof(numbers) / sizeof numbers[0];
printf("the length of the array: %u \n", length);
```

Output:

```
[choang7@gsuad.gsu.edu@snowball Lab9]$ ./addressOfArray
numbers = 0x7ffd66f48aa0
numbers[0] = 0x7ffd66f48aa0
numbers[1] = 0x7ffd66f48aa4
numbers[2] = 0x7ffd66f48aa8
numbers[3] = 0x7ffd66f48aac
numbers[4] = 0x7ffd66f48ab0
sizeof(numbers) = 20
the length of the array: 5
[choang7@gsuad.gsu.edu@snowball Lab9]$
```