

Lab Manual 1 – Pointers, Filing and Debugging

Important Note:

- You may find the syntax to accomplish these exercises from lecture demo.
- Names of your submission files should start with your roll number throughout this semester.
- Make sure that the interface of your program is user friendly i.e. properly display information.
- Properly follow the coding standards.

1 Exercise – Debugging [30 Minutes]

See the following piece of code and write its output by debugging the code

```
int myFunction ()
{
    int numbers[5];
    int * p;
    p = numbers;
    *p = 10;
    p++;
    *p = 20;
    p = &numbers[2];
    *p = 30;
    p = numbers + 3;
    *p = 40;           p
    = numbers;
        *(p+4) = 50;

    for (int n=0; n<5; n++)
        cout << numbers[n] << " ";

    return 0;
}
Void main()
{ myFunction();
}
```

Write the address of array named 'numbers'

0	1	2	3	4
0x00bffdcc8	0x00bffdcc	0x00bffd0	0x00bffd4	0x00bffd8










Sr. No	code	Value of p	Address of p	Value of array 'numbers'				
				[0]	[1]	[2]	[3]	[4]
1	int numbers[5];	???	0x00bffd0	- 85899346 0	- 8589 9346 0	- 8589934 60	- 8589934 60	- 85899346 0
2	int * p=numbers;	0x00bffdcc8	0x00bffdcc8	- 85899346 0	- 8589 9346 0	- 8589934 60	- 8589934 60	- 85899346 0
3	*p = 10;	0x00bffdcc8	0x00bffdcc8	10	- 8589 9346 0	- 8589934 60	- 8589934 60	- 85899346 0
4	p++;	- 8589 9346 0	0x00bffdcc	10	- 8589 9346 0	- 8589934 60	- 8589934 60	- 85899346 0
5	*p = 20;	0x00bffdcc	0x00bffdcc	10	20	- 8589934 60	- 8589934 60	- 85899346 0
6	p = &numbers[2];	-858993460	0x00bffd0	10	20	- 8589934 60	- 8589934 60	- 85899346 0

7	*p = 30;	0x00bffd0	0x00bffd0	10	20	30	- 858993460	- 858993460
8	p = numbers + 3;	- 8589934	0x00bffd4	10	20	30	- 858993460	- 858993460
							60	0
9	*p = 40;	0x00bffd4	0x00bffd4	10	20	30	40	- 858993460
10	p = numbers;	0x00bfddc8	0x00bffd4	10	20	30	40	- 858993460
11	*(p+4) = 50;	0x00bfdd8	0x00bffd8	10	20	30	40	50

Help: Debugging commands:



Short cut key	Icon	Menu	Explanation
---------------	------	------	-------------

F-9			Insert/Remove breakpoint
F-5		Debug-Go	Execute a program until the next breakpoint
Shift F-5		Debug-Stop debugging	To stop debugging a program. It will stop executing the program
F-10		Debug-StepOver	Go to the next statement
F-11		Debug-Step Into	Go inside a function
Shift F-11		Debug – Step Out	Come out of the function
		Debug - Run to cursor	Execute all statements till the statement on which the cursor is placed or until the next breakpoint
Alt -3		Debug-Windows-Watch	Show the window where only the variables in scope are shown
Alt-4		Debug-Windows-Variables	Show the window in which you can type a variable name to see its value
Alt-7		debug-windows-call stack	You can see the activation of stack of functions here

2 Exercise – Filing [15 Minutes]

Follow these steps:

- 1- Create a new cpp file, “filing.cpp”.
- 2- Create a text file “input.txt” in the current directory.
- 3- Paste following data in “input.txt”

```
10
21 62 53 94 51 26 97 88 39 111
5
94 73 28 44 95
```

- 4- Paste following code in filing.cpp, execute and check what it does. (To be demonstrated by the Lab Instructors)

```

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

void main()
{   ifstream fin("input.txt"); //input file stream, it will work on file
input.txt. Make sure your current cpp file and input.txt are in same folder

    if(fin.is_open()) //Checking if file has been opened
    {
        //If file has been opened, process it otherwise display error message

        int size1=0;
        int size2=0;
        int number = 0;

        fin>>size1;          //reading first integer from file

        cout<<"First Array:\n";
        cout<<"Size:\t"<<size1<<endl; cout<<"Data:\t";

        for(int i=0 ; i<size1 ; i++)
        {
            //This loop is reading total 10 numbers, where 10 is Size1

            fin>>number; //Read next integer from file
            cout<<number<<" ";
        }

        fin>>size2;          //reading next integer from file

        cout<<"\n\nSecond Array:\n";
        cout<<"Size:\t"<<size2<<endl; cout<<"Data:\t";

        for(int i=0 ; i<size2 ; i++)
        {
            //This loop is reading total 5 numbers, where 5 is Size2

            fin>>number; //Read next integer from file
            cout<<number<<" ";
        } cout<<endl; fin.close(); //Close the file as soon as you
        finish reading it.
    }
}

```

```

    }
    else
    {
        cout<<"Cannot open file.\n";
    }
}

```

3 Exercise – Basic Pointers [30 Minutes]

Follow these steps:

1. Declare:
 - a. Int variables num1, num2 and sum
 - b. Int* pointer variables xPtr, yPtr and sumPtr
2. Set num1, num2 and sum to 5, 7 and 0 respectively
3. Initialize all pointers to 0 (nullptr)
4. Print values of variables num1, num2 with labels as shown in the required output below: **Required**

Output:

```

Num1 = 5
Num2 = 7

```

5. Print the addresses of Num1 and Num2 using Address Operator (&). Required Output (assuming addresses starting from 0x10 for Num1 and so on) is shown below. (Note that addresses will be different on different machines)

Required Output:

```

Num1 = 5
Num2 = 7
Address of Num1 = 0x10 //(This will be different on your machine)
Address of Num 2 = 0x14 //(This will be different on your machine)

```

6. Point xPtr to num1 and yPtr to num2
7. Print values of Num1 and Num2 by dereferencing xPtr and yPtr **Required Output:**

```

Num1 = 5
Num2 = 7
Address of Num1 = 0x10 //(This will be different on your machine)
Address of Num 2 = 0x14 //(This will be different on your machine)
*xPtr = 5
*yPtr = 7

```

8. Point sumPtr to sum and print sum by dereferencing sumPtr. **Required Output:**

```
Num1 = 5
Num2 = 7
Address of Num1 = 0x10 //(This will be different on your
machine) Address of Num 2 = 0x14//(This will be different on
your machine) *xPtr = 5 *yPtr = 7
*sumPtr = 0
```

9. Add num1 and num2 using *xPtr and *yPtr and save the result in integer sum 10. Again Print sum using sumPtr **Required Output:**

```
Num1 = 5
Num2 = 7
Address of Num1 = 0x10 //(This will be different on your
machine) Address of Num 2 = 0x14//(This will be different on
your machine) *xPtr = 5 *yPtr = 7
*sumPtr = 12
```

11. Print the values of xPtr and yPtr (cout<<"xPtr = "<<xPtr<<endl) **Required Output:**

```
Num1 = 5
Num2 = 7
Address of Num1 = 0x10 //(This will be different on your machine)
Address of Num 2 = 0x14//(This will be different on your machine)
*xPtr = 5
```

```
*yPtr = 7 *sumPtr = 12 xPtr = 0x10 //This output should be same as
address of num1 i.e. &num1 yPtr = 0x14 //This output should be same as
address of num2 i.e. &num2
```

Help:

```
cout<<"Num1 = "<<num1<<endl; // Prints Num1 = 5
sum = *xPtr + *yPtr // Add num1 and num2 using *xPtr and *yPtr and save the result in integer sum
```

4 Exercise – Expand Array [30 Minutes]

Write a program that keeps taking integer input from the user until user enters -1 and displays the data in reverse order.

Your program should save the input in a dynamically allocated array. Initially create a dynamic array of five integers. Each time the array gets filled your program should double the size of array (i.e. create a new array of double size, copy previous data in new array, delete previous array) and continue taking the input. After receiving -1 (i.e. end of data input) your program should print the numbers in the reverse order as entered by the user.

Important Note: subscript operator [] is not allowed to traverse the array. Use only offset notation. i.e instead of using myArray[i] use *(myArray+i) to read/write an element. **Do not consume extra space. There shouldn't be any memory leakage or dangling pointers in your code.**

5 Practice Problems

These problems are only for your practice, their submission is not required.

Important Note: Do not consume extra space. There shouldn't be any memory leakage or dangling pointers in your code. **You ARE ALLOWED to use subscript notation (e.g. myArrayPtr[i]) to read/write ith index and integer iterators ARE ALLOWED in loop (i.e. (for int i = 0; i < size; i++)).**

5.1 Question 1 - Compress Array:

Write a program that takes size of an array and its elements and removes consecutive occurrences of same number from the list. For the example given below, your program should have space of exactly 7 integers on heap after compression. Do not consume any extra byte on heap.

Sample Run:

Array Before Compression: 1,1,2,2,2,3,4,5,5,5,5,7,7,7,2,2,2

Array After Compression: 1,2,3,4,5,7,2

5.2 Question 2 - Find Common Elements:

Implement a function that finds common elements in two sorted arrays. If array1 = {1,2,3,4,5,6} and array2 is {1,3,5,7}, then array3 (common elements) should be {1,3,5}. Note array3 should not have any duplicate elements and at the end array3 should not consume a single extra byte on heap. You have to:

- Allocate the three arrays dynamically after inputting the size of array1 and array2 from the user. Statically allocated arrays are NOT allowed.
- Initially you can allocate elements = (size of array1 + size of array2) to array3. For example you would allocate 6+4 to array 3 for the above example. After finding the common elements, the allocated size of array3 may be more than what you need. (In the above example you require 3 whereas you have allocated 12).