

## Goal

Practice the pointers and dynamic memory allocation

## Input to the program

- Input is internal: code in *main()* function is used to test the functionality

## Procedure for the implementation

- There are four problems in this assignment.
- Each problem is separate and must be implemented in a separate cpp file

## General remarks

- Keep all your testing code in submitted cpp files
- For all the problems, ensure/add the proper memory allocation/deallocation (all instructions about memory are not necessarily mentioned in the instruction).
- For all the problems, please use **valgrind** tool to confirm the proper memory management. Use the command:

```
valgrind --tool=memcheck --leak-check=yes --show-reachable=yes --  
num-callers=20 --track-fds=yes ./01.o
```

where **01.o** is the name of tested binary file

## Problem 1. Simple pointers usage (20%)

1. Write a program, that will be reading double values into the dynamic array
  - Ask user how many numbers user wants to store in the array, store this number in **asize**
  - declare two pointers of double type: *\*p\_min*, *\*p\_max*
  - read numbers in the loop, that has the following steps:
    - read a number into array
    - update *\*p\_min* and *\*p\_max*, to point to minimum and maximum elements in the array. If either *\*p\_min* or *\*p\_max* are updated, then write: "new minimum value at address **newaddress**, now pointing to value: **value**"
    - display:
      - "iteration number: " *number of current iteration*
      - "read number: " *number that was read in current iteration*
      - "current array elements: " *all array elements, separated by commas*
      - "current minimum = " *value of double pointed by \*p\_min* "at address: " *address stored in \*p\_min*
      - "current maximum = " *value of double pointed by \*p\_max* "at address: " *address stored in \*p\_max*
  - start numbering of iterations from 0

Example output for single iteration: (assuming, that already read 5 elements: 7, 15, 2, 3 and 11):

```
Iteration number: 5
Please enter element number 5: 19
Updating p_max to address 0x69583867, now pointing to value: 19
Read number: 19
Current array elements: 7,15,2,3,11,19
Current minimum = 2 at address 0x695834
Current maximum = 19 at address 0x6958387F
```

## Problem 2. Accessing classes with pointers (25%)

Steps to implement the program: (add other steps if necessary).

- Declare class *Student*, with the following public members: *name(string)*, *id(int)*, *gpa(double)*
- Create dynamic array *sArr* of *Student* objects, size of the array is entered by from keyboard (like in problem 1)
- Declare pointer *\*current* of type *Student*
- Read all the student data from the keyboard. Access the array element using *\*current* pointer (e.g. before writing to the array element, assign this element's address to *\*current* and access the element through *\*current* pointer only).
- Declare *Student* object named *\*BestGpaOfSemester* using *new* operator
- Declare *Student* object named *\*LowestGpaOfSemester* using *new* operator
- Iterate over the *sArr* and find the student with the highest *gpa*. Copy his data into *\*BestGpaOfSemester* object (i.e. when values in *sArr* change, the value in *\*BestGpaOfSemester* does not change).
- Iterate over the *sArr* and find the student with the lowest *gpa*. Copy his data into *\*LowestGpaOfSemester* object (i.e. when values in *sArr* change, the value in *\*LowestGpaOfSemester* does not change).
- Print the whole content of the *sArr*, one record per line. Use [] to access elements, include address of the array and address of each element.
- Delete the *sArr* array
- Print the *\*BestGpaOfSemester* student (include address of the object)
- Print the *\*LowestGpaOfSemester* student (include address of the object)

Example output:

```
Please input the size of the array: 4
Enter student 1 name: _____
Enter student 1 id: _____
Enter student 1 gpa: _____
Enter student 2 name: _____
Enter student 2 id: _____
Enter student 2 gpa: _____
Enter student 3 name: _____
Enter student 3 id: _____
Enter student 3 gpa: _____
Enter student 4 name: _____
Enter student 4 id: _____
Enter student 4 gpa: _____

Student list: (array address: 0xd96cc5a0)
Student 1: John Smith, 2000838696, 3.66 (at address 0xd96cc5a0)
Student 2: Anna White, 1000436353, 3.80 (at address 0xd96cc600)
Student 3: Paul Doe, 2000222423, 3.00 (at address 0xd96cc660)
Student 4: Mary Green, 2000281323, 3.89 (at address 0xd96cc6c0)
```

Array deleted.

Student with highest GPA:

Anna White, 1000436353, 3.80 (at address 0xd96cc6e2)

Student with lowest GPA:

Paul Doe, 2000222423, 3.00 (at address 0xd96cc6f6)

### Problem 3. Pointers arithmetic (35%)

Steps to implement the program:

- Ask the user, about the *int* array size, store it in **asize** variable
- Create the dynamic array of integers **DArr**, with the size of *asize*
- Fill the array: *n-th element receives the value  $n*(n+1)$*
- Print the whole array, separate elements with commas
- Declare pointer *\*current*, point it to the first element of the array
- Create infinite loop. During each loop iteration, do the following:
  - Ask the user, which **relative** element user wants to access. This means, that you are asking user: how many positions to the left or to the right of the current element, is the element you want to jump to located?
  - Each time user enters location outside of the range, set *\*current* to the first element of the array

Example: (*\*current* points to the first element of the array)

- Array contents: 0,2,6,12,20,30,42,56,72,90,110
  - User enters: 5
  - Program prints (in one line): address of the **DArr**, address of *\*current* element, address of the last element
  - Program prints the element of index 5: i.e. 6<sup>th</sup> element to the right from the first element in the array: **30**, program prints: “current value=30, current position: 5/9” (x/y: x=position of the current element (index of the element) – calculate using array address and \*current address, y = *asize*)
  - User enters: -2
  - Program prints (in one line): address of the **DArr**, address of *\*current* element, address of the last element
  - Program prints: “current value=12, current position: 3/9”
  - User enters: **30**
  - Program prints (in one line): address of the **DArr**, address of *\*current* element, address of the last element
  - Program prints: *Position set to 0.*
  - Note that you have to update the pointer *\*current* each time user enters the value
- Quit your program, when user enters -1111
- Note, that you **must** use the addresses to calculate the integer position of *\*current* (“current position”, also denoted as x above). This is “pointer arithmetic”. You can’t have separate variable, where you store the position. Use *addr\_1 – addr\_2* to calculate the offset.

Sample output:

enter array size: 10  
0, 2, 6, 12, 20, 30, 42, 56, 72, 90,  
enter relative pos: 5

DArr=0x1810010 current=0x1810024 last=0x1810038

current value=30, current position=5/9

enter relative pos: -2

DArr=0x1810010 current=0x1810016 last=0x1810038

current value=12, current position=3/9

enter relative pos: 30

Position set to 0.

current pos = 0

new pos = 0 + 5 = 5

current pos = 5

new pos = 5 - 2 = 3

current pos = 3

current pos = 0

#### Problem 4. Pointers in classes (20%)

Write a program:

- Declare a class *DArr*, with the following members:
  - public:
    - double \*arr // pointer to the array
    - int size // size of the array
    - void displayElements() // displays all elements of the array, uses \* operator
    - double\* getMax() // returns the pointer to maximum number from the array
    - DArr() // constructor (see description below)
    - ~DArr() // destructor (see description below)

Constructor *DArr()*:

- Asks user for the size of the array
- Dynamically creates the array **arr** of the size provided
- Prints “Constructor: allocating X bytes of the memory” (calculate X knowing the size of *double* and number of elements in the array)
- Fills the array with random *double* values, from the range 0..1000. Include fraction part in the randomization process.
- For calculations, use *sizeof()* function to determine the size of single element

Destructor *~DArr()*:

- Prints: “Destructor: freeing X bytes of the memory” (calculate X knowing the size of *double* and number of elements in the array)
- Deallocates the memory allocated by constructor
- The *displayElements()* function **must** access elements using *\*( )* operator, do not use *[]* operator.
- The *getMax()* function **must** access elements using *\*( )* operator, do not use *[]* operator.

Test:

- Create object d1 of *DArr*
- Display all the elements of *DArr* using *displayElements()* function
- Get the maximum element of *DArr* using *getMax()* function

## Test 2:

- In *main()* function, create an array of *DArr* objects (prompt user for its size)
- Display all elements from the array, for each element of the array display all the double values, along with max double

## Sample output:

Enter the array size: 20

Constructor: allocating 160 bytes of memory

array elements:

311.23,751.60,16.11,224.53,813.89,497.33,199.64,832.93,846.21,195.90,429.22,508.34,684.29,30.56,574.12,116.54,997.55,157.89,667.63,614.14

max element: 997.55 at the address 0x0398493

Destructor: freeing 160 bytes of memory