Laboratory 09A

CS-102

Spring 2022

Laboratory 9A Part 1

Dynamic Arrays – Creating Arrays on the Fly

- When C++ was first created as a programming language, Arrays were created as static structures at the beginning of a program and could never be altered.
- Two enhancements have been made to C++ which allow us the opportunity to create Arrays on the fly, similar to what we routinely do using Vectors.
- The first of these enhancements is what is called: Dynamic Memory Allocation.
- You'll find an example of Dynamic Memory Allocation in Program 9-14.
- Type in Program 9-14 and get it working.
- Notice that at the end of the program we need to do a little housekeeping by deleting the Dynamic Array that we have created, and reassigning its pointer name to nullptr. We do this to prevent "memory leaks."

Program 9.14 page 1 of 2

Use of Dynamic Memory Allocation to create new Array on the Fly

```
// This program totals and averages the sales figures for any
// number of days. The figures are stored in a dynamically
// allocated array.
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
   double *sales = nullptr,                  // To dynamically allocate an array
   total = 0.0,                              // Accumulator
   average;                                  // To hold average sales
   int numDays,                              // To hold the number of days of sales
       count;                                // Counter variable
   // Get the number of days of sales.
   cout << "How many days of sales figures do you wish ";
   cout << "to process? ";
   cin >> numDays;

   // Dynamically allocate an array large enough to hold
   // that many days of sales amounts.
   sales = new double[numDays];
```

```
        // Get the sales figures for each day.
        cout << "Enter the sales figures below.\n";
        for (count = 0; count < numDays; count++)
        {
                        cout << "Day " << (count + 1) << ": ";
                        cin >> sales[count];
        }
        // Calculate the total sales
        for (count = 0; count < numDays; count++)
                        total += sales[count];
        // Calculate the average sales per day
        average = total / numDays;
        // Display the results
        cout << fixed << showpoint << setprecision(2);
        cout << "\n\nTotal Sales: $" << total << endl;
        cout << "Average Sales: $" << average << endl;
        // Free dynamically allocated memory
        delete [] sales;
        sales = nullptr; // Make sales a nullptr.
        return 0;
}
```

Program 9.14 page 2 of 2

Use of Dynamic Memory Allocation to create a new Array on the Fly

Laboratory 9A - Part 2
Dynamic Arrays – Creating Arrays on the Fly

- In 2011, the C++ compiler was upgraded to C++ 11 and Smart Pointers were introduced.
- These had the ability to delete Dynamically Allocated Memory that was no longer needed.
- This means that you no longer needed to do the housekeeping of deletion and reallocation that you needed to do with Dynamically Allocated Memory.
- Program 9-18 shows an example of the use of Smart Pointers with an Array. Type this program in and get it working.
- Note that use of Smart Pointers requires the use of the <memory> library.

```
// This program demonstrates a unique_ptr pointing
// to a dynamically allocated array of integers.
#include <iostream>
#include <memory>
using namespace std;

int main()
{
        int max;   // Max size of the array

        // Get the number of values to store.
        cout << "How many numbers do you want to enter? ";
        cin >> max;

        // Define a unique_ptr smart pointer, pointing
        // to a dynamically allocated array of ints.
        unique_ptr<int[]> ptr( new int[max]);
```

Program 9.18 page 1 of 2

Use of Smart Pointers to create new Array on the Fly

```
    // Get values for the array.
    for (int index = 0; index < max; index++)
    {
      cout << "Enter an integer number: ";
      cin >> ptr[index];
    }

    // Display the values in the array.
    cout << "Here are the values you entered:\n";
    for (int index = 0; index < max; index++)
      cout << ptr[index] << endl;

    return 0;
}
```

Program 9.18 page 2 of 2

Use of Smart Pointers to create a new Array on the Fly

Laboratory 9A - Part 3
Dynamic Arrays – Creating Arrays on the Fly

- Now we'll upgrade Program 9.14 so that it will use Smart Pointers instead of Dynamic Memory Allocation.
- Rewrite Program 9.14 so that it works using Smart Pointers instead of Dynamic Memory Allocation.
- Call your program: *YourName*-Lab09A-3.cpp
- When you have it working:
  - If you are doing this Lab synchronously, call the Instructor over so that you can be given proper credit for your work.
  - If you are doing this Lab asynchronously, submit your program to Canvas.

Laboratory 9A - Part 4
Dynamic Arrays – Creating Arrays on the Fly

- Finally, let's rewrite Program 9-18 so that it works with Dynamically Allocated Memory, rather than Smart Pointers. (You might need to do this if you do not have access to the C++11 compiler).
- Call your program: *YourName*-Lab09A-4.cpp
- Once you have Program 9-18 modified so that it no longer needs the <memory> library:
  - If you are doing this Lab synchronously, call the Instructor over so that you can be given proper credit for your work.
  - If you are doing this Lab asynchronously, submit your program to Canvas.

Laboratory 9A - Part 5

Dynamic Arrays – Creating Arrays on the Fly

- It is possible (although not likely) that **new** cannot find enough memory to allocate. It would return a Bad Allocation Error. [Note this would also apply to Smart Pointers.] Good programming practice means you test for this, using a try/catch construct before proceeding with your program.
- Suppose, you were creating a **new** array of int of size 1,000,000,000:

```
int *ptr;
long long size=1000000000;
try
{
        ptr = new int[size];
}
catch (const bad_alloc &)
{
        cout << “Error allocating memory”;
        exit(0);
}
…  // continue on with program
cout << “Success!”;
```

Laboratory 9A - Part 5
Dynamic Arrays – Creating Arrays on the Fly

- What we're going to do in this section is to find the limits of how large a Dynamic Array your particular machine configuration can generate before it "gives up."
- Your program will do the following:
  - It will ask how big an array you would like to make.
  - Then it will attempt to make an array of that size.
  - If the array is successfully made, then it will display the word "Success!"
  - Otherwise, it will display the words "Error Allocating Memory"
  - Using this program, determine to four decimal places just how large a Dynamic Array your machine will create before it quits.
- Do this exercise for both arrays of int as well as arrays of double.
  - Call your array of int: *YourName*-Lab09A-5int.cpp
  - Call your array of double: *YourName*-Lab09A-5double.cpp

Laboratory 9A - Part 5
Dynamic Arrays – Creating Arrays on the Fly

- If you are doing this Lab synchronously, show the instructor your code as well as the results of your test.
- If you are doing this Lab asynchronously, submit your results for an array of int, and an array of double with a text file, as well as *YourName*-Lab09A-5int.cpp and *YourName*-Lab09A-5double.cpp

Laboratory 9A - Part 6
Dynamic Arrays – Creating Arrays on the Fly

- We will now repeat what we did in part 5, but this time using Smart Pointers and creating an array of char, rather than an array of int.
- Your program will do the following:
  - It will ask how big an array you would like to make.
  - Then it will attempt to make an array of that size.
  - If the array is successfully made, then it will display the word “Success!”
  - Otherwise, it will display the words “Error Allocating Memory”
  - Using this program, determine to four decimal places just how large a Smart Pointer can create an array on your machine before it quits.
- Do this exercise for arrays of char.
  - Call your array of char: *YourName*-Lab09A-6char.cpp

Laboratory 9A - Part 6
Dynamic Arrays – Creating Arrays on the Fly

- Based on your experimental data, approximately how much “heap” memory do you have available for C++ on your computer.
- If you are doing this Lab synchronously, show the instructor your code as well as the results of your test.
- If you are doing this Lab asynchronously, submit your results for an array of char, as well as *YourName*-Lab09A-6char.cpp.