# EN 001203 Computer Programming

## L007: Arrays

**Faculty of Engineering, Khon Kaen University**

Submission: https://autolab.en.kku.ac.th

==========

* Submit an answer to a question with a file with `txt` extension. E.g., an answer for Q1 should be submitted in a text file "`Q1.txt`"

* Submit a program to a (programming) problem with a file with `cpp` extension. E.g., a program for P2 should be named "`P2.cpp`"

* All answers and programs must be packaged together in a tar file.

* Each question or problem is worth 60 points. The 240 points are counted as full score. Students are encouraged to work on as many problems as they can, but 240 points should be adequate.

==========

"Talent wins games, but teamwork and intelligence win championships."

--- Michael Jordan

P1. Simple array. Write a program to take (floating-point) values from a user and put it in an array.

Use a start code below.

Starter code:

```
#include <iostream>

using namespace std;

int main(){
      int N;
      cout << "N: ";
      cin >> N;

      float a[N];

      // Write your code here!!!
```

```
	cout << "Your list is [";
	cout << a[0];
	for(int i=1; i < N; i++){
		cout << ", " << a[i];
	}
	cout << "]" << endl;

	return 0;
}
```

Have the program interact exactly like what shown in the examples. A number of elements and element values are inputted by a user.

Output example: when inputting 3 (for a number of elements), then 7, 2, and 4 for values.

```
N: 3
0: 7
1: 2
2: 4
Your list is [7, 2, 4]
```

P2. Simple array and function. Write a function to compute a summation of all values of the array elements. The function takes an array and its size and returns its summation. Make a function interface

```
float asum(float a[], int size)
```

[Hint: a function argument of a 1D array type can be declared as `*a` or `a[]` or `a[#]` with # represents an array size (e.g., `a[5]` for argument of a length-5 array).]

Use a start code below.

Starter code:

```
#include <iostream>

using namespace std;

// Write your code here!!!

int main(){

	int N;
	cout << "N: ";
	cin >> N;
```

```
	float a[N];

	for(int i=0; i < N; i++){
		cout << i << ": ";
		cin >> a[i];
	}

	cout << "sum = " << asum(a, N) << endl;

	return 0;
}
```

Have the program interact exactly like what shown in the examples. A number of elements and element values are inputted by a user.

Output example: when inputting 3 (for a number of elements), then -2.5, 5, and 3.2 for values.

```
N: 3
0: -2.5
1: 5
2: 3.2
sum = 5.7
```

P3. Vector as an array. Write a function to compute a vector addition. The function takes two vectors (each as an array), their sizes, and a dummy array for the result. The function returns 0 if both sizes are matched and -1 otherwise. Make a function interface

```
int vadd(float v[], float u[], int Nv, int Nu, float r[])
```

[Hint: a function argument as `*a` or `a[]` are reference, i.e., their value change will be observed beyond the scope of its function.]

Use a start code below.

Starter code:

```
#include <iostream>

using namespace std;

int vadd(float v[], float u[], int Nv, int Nu, float r[]){
	// Write your code here!!!
}

int getvec(float v[], int size){
	for(int i=0; i < size; i++){
		cout << i << ": ";
```

```
            cin >> v[i];
      }
      return 0;
}

int dispvec(float v[], int size){
      cout << "[";
      for(int i=0; i < size-1; i++){
            cout << v[i] << ", ";
      }
      cout << v[size-1] << "]";
      return 0;
}

int main(){

      int size1, size2;
      cout << "V1 size: ";
      cin >> size1;
      cout << "V2 size: ";
      cin >> size2;

      float v1[size1], v2[size2];
      float a[size1];

      cout << "V1:" << endl;
      getvec(v1, size1);
      cout << "V2:" << endl;
      getvec(v2, size2);

      if( vadd(v1, v2, size1, size2, a) == 0){
            cout << "Addition = ";
            dispvec(a, size1);
      }else{
            cout << "Vectors do not meet a dimension agreement.";
      };
      cout << endl;

      return 0;
}
```

Have the program interact exactly like what shown in the examples. Vector sizes and values are inputted by a user.

Output example: when inputting 3 and 3 for both vector sizes, then 1,5,8 for values of vector V1 and -3,0,2 for values of vector V2.

```
V1 size: 3
V2 size: 3
V1:
0: 1
1: 5
```

```
2: 8
V2:
0: -3
1: 0
2: 2
Addition = [-2, 5, 10]
```

Output example: when inputting 2 for V1 size and 3 for V2 size, then 4, 2 for values of vector V1 and 1, 5, 3 for values of vector V2.

```
V1 size: 2
V2 size: 3
V1:
0: 4
1: 2
V2:
0: 1
1: 5
2: 3
Vectors do not meet a dimension agreement.
```

P4. Matrix as a 2D array. Write a program to take values of a 3x3 matrix.

[Hint: a matrix can be represented by a 2D array.]

Use a start code below.

Starter code:

```
#include <iostream>

using namespace std;

int dispmat(float m[3][3]){
      for(int i=0; i<3; i++){
            for(int j=0; j<3; j++){
                  cout << m[i][j] << " ";
            }
            cout << endl;
      }
      return 0;
}

int main(){

      float m[3][3];

      cout << "Matrix:" << endl;
      // Write your code here!!!
```

```
	cout << endl << "m =" << endl;
	dispmat(m);

	return 0;
}
```

Have the program interact exactly like what shown in the examples. Matrix elements are inputted by a user.

Output example: when inputting 1, 2, 3, 5, 6, 4, -1, 0, and -3.

```
Matrix:
1 2 3
5 6 4
-1 0 -3

m =
1 2 3
5 6 4
-1 0 -3
```

P5. Matrix multiplication. Write a function to compute a matrix multiplication for 2x2 matrices. The function takes multiplicand and multiplier matrices as well as a resultant matrix. The function returns 0. Make a function interface

```
int matmult(float m1[2][2], float m2[2][2], float r[2][2])
```

[Hint: $r_{ij} = \sum_{k=1}^{K} a_{ik} \cdot b_{kj}$, where $r_{ij}$ is the (i, j) element of the resultant matrix, $a_{ik}$ is the (i,k) element of the multiplicand and $b_{kj}$ is the (k,j) element of the multiplier. You may need 3 nested loops: one for i, one for j, and another one for k.]

Use a start code below.

Starter code:

```
#include <iostream>

using namespace std;

int dispmat(float m[2][2]){
	for(int i=0; i<2; i++){
		for(int j=0; j<2; j++){
			cout << m[i][j] << " ";
		}
		cout << endl;
```

```
      }
      return 0;
}

int getmat(float m[2][2]){
      cout << "Matrix:" << endl;
      cin >> m[0][0] >> m[0][1];
      cin >> m[1][0] >> m[1][1];
      return 0;
}

int matmult(float m1[2][2], float m2[2][2], float r[2][2]){

      // Write your code!!!

      return 0;
}

int main(){

      float A[2][2], B[2][2], C[2][2];

      getmat(A);
      cout << endl;
      getmat(B);
      cout << endl;

      matmult(A, B, C);

      cout << "Product= " << endl;
      dispmat(C);

      return 0;
}
```

Have the program interact exactly like what shown in the examples. Values are inputted by a user.

Output example: when inputting 1, 4, 0, -1 for the multiplicand and 4, 2, 0.5 and 0.2 for the multiplier.

```
Matrix:
1 4
0 -1

Matrix:
4 2
0.5 0.2

Product=
```

```
6 2.8
-0.5 -0.2
```

"If you want to build a ship, don't herd people together to collect wood and don't assign them tasks and work, but rather teach them to long for the endless immensity of the sea."

--- Antoine de Saint-Exupery