

Lab 3.10.2 Dynamic data – how to obtain it and how to get rid of it

Objectives

Familiarize the student with:

- assigning portions of data with **new** and disposing of it with the **delete** operator;
- the concept of dynamic data collection.

Scenario

Look at the code below – it's a skeleton of a program operating on the dynamic collection of data. The idea is to use a structure containing two fields: the first stores the number of elements in collections, and the second is the actual collection (a dynamically allocated vector of ints). As you can see, the collection is filled with the required amount of pseudo-random data.

Unfortunately, the program requires completion, as the most important function, intended to add elements to the collection, is still empty. Are you ready to take up the challenge?

Here's what we expect from the function:

- if the collection is empty, it should allocate a one-element vector and store a new value in it;
- if the collection is not empty, it should allocate a new vector with a length greater by one than the current vector, then copy all elements from the old vector to the new one, append a new value to the new vector and finally free up the old vector.

Run your code several times and make sure that everything works as it should.

```
#include <iostream>
#include <cstdlib>
#include <ctime>

using namespace std;

struct Collection {
    int elno;
    int *elements;
};

void AddToCollection(Collection &col, int element) {

    // Insert your code here

}

void PrintCollection(Collection col) {
    cout << "[ ";
    for(int i = 0; i < col.elno; i++)
        cout << col.elements[i] << " ";
    cout << "]" << endl;
}

int main(void) {
    Collection collection = { 0, NULL };

    int elems;
    cout << "How many elements? ";
    cin >> elems;
    srand(time(NULL));
    for(int i = 0; i < elems; i++)
        AddToCollection(collection, rand() % 100 + 1);
    PrintCollection(collection);
    delete[] collection.elements;
    return 0;
}
```