# Programowanie 1

## Zadanie 8

Piotr Błaszyński

23 listopada 2021

Zmodyfikować funkcje z programu 7 (tablice), tak aby przyjmowały i zwracały tablice alokowane dynamicznie. Wykorzystać do tego funkcje malloc. Jeżeli tablica jest zwracana z funkcji, to należy ją zwolnić przez free po skorzystaniu z z tej tablicy.

Sztuczka w debugerze VS (i nie tylko): jeśli w okienku Watch jako nazwę podglądanej zmiennej napiszemy data,15 to zobaczymy podgląd 15 elementów pamięci (w naszym przypadku liczb rzeczywistych, ale działa również dla obiektów).

Przykładowe wykorzystanie dynamicznie alokowanych tablic:

```
#include <stdlib.h>
#include <stdio.h>
#define VALUES_SIZE 10
void foo()
{
  int *values;
  values=(int *)malloc(sizeof(int)*VALUES_SIZE);
  for (int i=0 ; i<VALUES_SIZE ; i++)
  {
    values[i]=i*2;//odwolanie jak do statycznych tablic
    *(values+i)+=3;//wykorzystanie arytmetyki wskaznikow
        //(na statycznych tablicach tez tak wolno)
  }
  for (int i=0 ; i<VALUES_SIZE ; i++)
        printf("value[%d] = %d\n", i, values[i]);
  free(values);
}
```

Przykładowe wykorzystanie tablic alokowanych dynamicznie z użyciem new[]/delete[] (to jest wersja dla C++, w przyszłości w razie konieczności można z niej skorzystać, w wielu wypadkach wystarczy jednak vector lub inna odpowiednia struktura danych):

```
#include <stdio.h>
#define VALUES_SIZE 10
void foo2()
{
  int *values;
  values=new int[VALUES_SIZE];//zwracam uwage na uzyte
     nawiasy[]
  for (int i=0 ; i<VALUES_SIZE ; i++)
  {
    values[i]=i*2;
    *(values+i)+=3;
  }
  for (int i=0 ; i<VALUES_SIZE ; i++)
    printf("value[%d] = %d\n", i, values[i]);
  delete [] values;//zwracam uwage na [], tak zwalniamy
     pamiec, jezeli jest kilka elementow
}
```

Przykładowe wykorzystanie dynamicznie alokowanych tablic ze zwracaniem ich z funkcji (tablicę zaalokowaną wewnątrz funkcji trzeba zwolnić na zewnątrz - potencjalne źródło błędów):

```
#include <stdlib.h>
#include <stdio.h>
float *allocEmptyValues(int count)
{
  float *values = (float *)malloc(sizeof(float)*count);
  for (int i=0 ; i<count ; i++)
    values[i]=0;
  return values;
}

#define VALUES_SIZE 10
void foo5()
{
  float *values=allocEmptyValues(VALUES_SIZE);
  for (int i=0 ; i<VALUES_SIZE ; i++)
    printf("value[%d] = %f\n", i, values[i]);
  free(values);
}
```