Programowanie obiektowe – Lista 6: wskaźniki i tablice

1. Program

```
4 int main()
  {
      double tab[10];
      double x;
      std::cout << &x << "\t" << &tab[-1] << "\n";
9 }
```

   produkuje w moim komputerze następujący wynik:

```
0x22ff18        0x22ff18
```

   Skomentuj i wyjaśnij to zjawisko. Jakie konsekwencje może mieć dla programu niepoprawne indeksowanie tablic?

2. Wyjaśnij, w jaki sposób w poniższym programie (napisanym w stylu C) wykorzystano indolencję programisty do włamania się do systemu. Czy działanie tego programu zależy od kompilatora, którym był kompilowany?

```
3 int main()
  {
      char haslo[8];    // tu będzie przechowywane hasło
6     char tmp[8];      // zmienna tymczasowa na wczytanie hasła
      strcpy(haslo, "Ta.jnE!"); // kopiuje drugi argument w miejsce pierwszego
      for ( ; ; )
      {
        std::cout << "podaj haslo: ";
11      std::cin >> tmp;
        if (strcmp(tmp, haslo) != 0) // czy teksty są takie same?
          std::cout << "przykro mi, haslo jest niepoprawne!\n";
        else
          break;
16    }
      std::cout << "witaj w systemie!\n";
  }
```

```
podaj haslo: 123456781234567
przykro mi, haslo jest niepoprawne!
podaj haslo: 1234567
witaj w systemie!
```

verte !

3. Zadeklaruj:

   (a) wskaźnik do tablicy 100 liczb typu `double`;
   (b) dwuwymiarową tablicę 5×5 wskaźników na `char`.
   (c) tablicę 3 wskaźników na `char`.
   (d) standardowy wektor wskaźników `void*`.
   (e) standardowy wektor o elementach typu „tablica czterech liczb typu `double`".
   (f) tablicę 5 adresów funkcji zwracających wartość typu `double` i pobierających jeden argument typu `double` przez wartość.

4. Zaimplementuj funkcję `double srednia(double tab[], int n)`, która oblicza średnią arytmetyczną `n` kolejnych elementów tablicy `tab`. Następnie zdefiniuj tablicę 10-elementową `double t[] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}` i przy pomocy swojej funkcji oblicz średnią

   (a) wszystkich elementów tablicy;
   (b) trzech pierwszych elementów tablicy;
   (c) elementów pomiędzy `tablica[3]` a `tablica[7]` (włącznie);
   (d) 20 elementów począwszy od `tablica[0]` (tu może wyskoczyć błąd podczas wykonywania programu, ale chodzi o to, by sprawdzić, że *kompilator* to „łyknie").

5. Napisz program, który co najmniej 10 milionów razy wywoła w pętli instrukcje `int* p = new int[100];` oraz `delete [] p;`. Na tej podstawie oszacuj czas działania pojedynczej pary tych instrukcji i porównaj z czasem potrzebnym Twojemu procesorowi na wykonanie pojedynczego dodawania.
   Wskazówka: w Linuksie czas działania programu można łatwo zmierzyć poleceniem `time`.

6. *Zadanie dla ambitnych, nieobowiązkowe*: Wygeneruj tablicę 1000000 liczb zmiennopozycyjnych o losowych wartościach z przedzialu $-1 \ldots 1$ i uporządkuj ją funkcją `qsort` wg rosnącej wartości bezwzględnej. Powodzenia!