Dynamiczna alokacja pamięci

Wskaźnik na int'a – pojedynczy int

Deklaracja (jako zmienna, jako pole... wszystko jedno):

```
int *wskaznik;
```

Dynamiczna alokacja pojedynczego inta (i późniejsze usunięcie):

```
wskaznik = new int;
// …
delete wskaznik;
```

Przypisywanie i odczytywanie wartości z zaalokowanego int'a (z użyciem wyłuskania):

```
*wskaznik = 123;
cout << *wskaznik << endl;
```

Wskaźnik na int'a – tablica dynamiczna

Deklaracja – tak samo, jak wyżej:

```
int *wskaznik;
```

Alokacja i usuwanie tablicy:

```
wskaznik = new int[10];
// …
delete[] wskaznik; // trochę inaczej – zwróćcie uwagę na nawiasy kwadratowe
```

Ważne: w momencie alokacji (new int[10]) tworzymy tablicę, która zawiera 10 nowych int'ów. Bezpośrednio po tej instrukcji wszystkie int'y są już utworzone, można z nich korzystać. Proszę zwrócić na to uwagę w kontekście dalszej części.

Korzystanie z elementów tablicy:

```
wskaznik[2] = 123;
```

Wskaźnik na wskaźnik na int'a – tablica int'ów tworzonych dynamicznie

Deklaracja:

int **wskaznik;

Taki wskaźnik może przechowywać **adres pojedynczego wskaźnika na int'a**, lub, co bardziej praktyczne, **tablicę wskaźników na int'y**.

Dynamiczna alokacja tablicy wskaźników na int'y:

wskaznik = new int*[10];

W tym momencie **nie mamy jeszcze żadnych int'ów**. Stworzyliśmy tablicę, składającą się z dziesięciu niezainicjalizowanych wskaźników.

Taką strukturę można wykorzystać na dwa sposoby. Po pierwsze, każdy z tych dziesięciu wskaźników może przechowywać adres na pojedynczego, dynamicznie zaalokowanego int'a. **Najczęściej właśnie w taki sposób pracuje się z tablicami dynamicznie alokowanych obiektów.**

```
for (int i = 0; i < 10; ++i)
{
	wskaznik[i] = new int;
}
```

Dopiero w tej pętli utworzyliśmy 10 rzeczywistych int'ów, które mogą przechowywać wartości liczbowe. Gdybyśmy nie była to tablica int'ów, tylko obiektów naszej klasy, to w trakcie wywołania pętli stworzylibyśmy 10 obiektów (i dla każdego wywołany zostałby jego konstruktor).

Po drugie, każdy z tych dziesięciu wskaźników może „przechowywać” dynamicznie zaalokowaną tablicę intów, co całościowo dawałoby nam tablicę dwuwymiarową:

```
for (int i = 0; i < 10; ++i)
{
	wskaznik[i] = new int[20];
}
```

W tej pętli stworzyliśmy łącznie 200 int'ów.

Trzeba pamiętać, żeby tak zaalokowaną pamięć zwalniać „od końca”, to znaczy najpierw w pętli usunąć zawartość każdego elementu naszej tablicy, a dopiero na koniec usunąć całą tablicę.

Referencje do wskaźników

Oprócz wskaźników na wskaźniki, możemy mieć też referencje do wskaźników. Przykładowo:

void funkcja(int *&referencja_do_wskaznika) …

void funkcja(int **&referencja_do_wskaznika_na_wskaznik) …

Działa to tak, jak referencja do każdego innego typu danych – funkcja ma możliwość zmodyfikowania przekazanej zmiennej (w tym przypadku może zmodyfikować adres przechowywany we wskaźniku).