Zmienne

SPIS TREŚCI

Na początek

Aby dodać do siebie dwie wartości wprowadzone z klawiatury, należy co najmniej pierwszą z nich gdzieś zapisać, aby dodać do wartości wczytanej później.

```
- wczytaj pierwszą wartość
- zapisz do pamięci jako wartość1
- wczytaj drugą wartość
- dodaj do niej wartość zapisaną w pamięci jako wartość1
- wyświetl sumę
```

Tak więc bez użycia zmiennych nie można w komputerze wykonać nawet najprostszego dodawania.

Definicja

Formalnie, zgodnie z podręcznikową definicją

> Zmienna jest obszarem pamięci komputera przeznaczonym do przechowywania pojedynczej wartości określonego typu.

Pamięć RAM

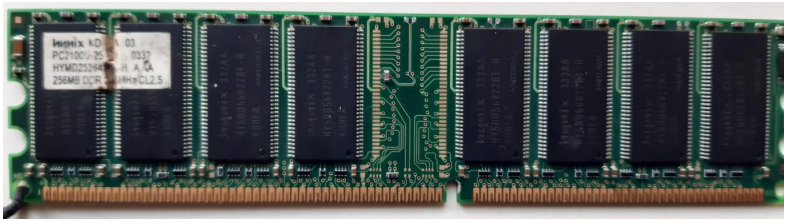
Faktycznie, w sensie fizycznym, elektrycznym zmienna jest więc fragmentem dostępnej pamięci (RAM), grupą kondensatorów których ładunek reprezentuje przechowywaną wartość.

Czym jest kondensator?

Kondensator jest prostym urządzeniem elektronicznym zbudowanym z dwóch metalowych blaszek rozdzielonych izolatorem, zwiniętych współosiowo[1]. Po podłączeniu tych blaszek do źródła napięcia gromadzi się na nich ładunek elektryczny, który pozostaje także po odłączeniu źródła napięcia.

Podstawowym zastosowaniem kondensatorów jest stabilizacja napięcia

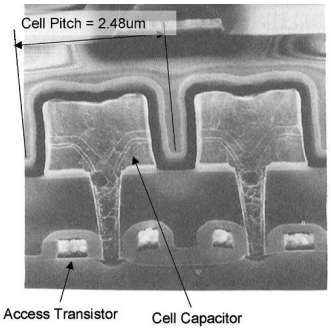
Czym jest pamięć RAM?



Pamięć RAM to ogromny zbiór bardzo małych kondensatorów, które można momentalnie ładować i rozładowywać. Co więcej z uwagi na fakt, że szybko tracą zgromadzony ładunek trzeba je nieustannie doładowywać.Na zdjęciu z mikroskopu elektronowego pokazany jest rzeczywisty wygląd kondensatorów obecnych na karcie pamięci RAM[2].

[1] https://en.wikipedia.org/wiki/Capacitor#/media/File:Capacitors_(7189597135).jpg

[2] https://www.researchgate.net/publication/3139122_Angular_dependence_of_multiple-bit_upsets_induced_by_protons_in_a_16_Mbit_DRAM/figures?lo=1



Wartości które przechowujemy w komputerze faktycznie są większym lub mniejszym zbiorem naładowanych i rozładowanych kondensatorów, z których każdy reprezentuje wartość jeden lub zero.

Jaką więc wartość reprezentuje określona grupa kondensatorów?

Wybrana grupa kondensatorów reprezentuje określoną wartość liczbową zapisaną w systemie dwójkowym. Na przykład w przypadku grupy ośmiu kondensatorów wartość binarna 1100 0101 odpowiada wartości 197 zapisanej w systemie dziesiętnym. Poza tym kondensatory niczego więcej nie reprezentują.

Typy zmiennych

Odczytane wartości z pamięci RAM podlegają interpretacji. Na przykład ten sam ciąg binarny może interpretować określoną wartość liczbową albo znak. Tak więc nie wystarczy zapisać wartości w pamięci komputera, konieczne jest także określenie i zapisanie typu wartości który konkretny "wpis" w pamięci reprezentuje.

W języku Java zdefiniowano osie tzw. typów podstawowych przeznaczonych do reprezentowania wartości różnych (podstawowych) typów.

1. **int** - od integer (całkowita)

   Typ przeznaczony do przechowywania wartości całkowitej za pomocą 4 bajtów, czyli 32 bitów, czyli 32 kondensatorów w pamięci RAM.

2. **double** - od double precision (czyli podwójna)

   Typ przeznaczony do przechowywania wartości w reprezentacji zmiennoprzecinkowej dającej możliwość zapisu części ułamkowej. Słowo podwójna precyzja odnosi się do wielkości pamięci przeznaczonej do przechowywania wartości

tego typu która w tym przypadku wynosi 8 bajtów, czyli 64 bitów.

3. **boolean** - od Georga Bool'a, pioniera logiki matematycznej

   Typ przeznaczony do przechowywania jednej z dwóch wartości logicznych — prawda (true) albo fałsz (false).

4. **char** - czyli character

   Typ przeznaczony do przechowywania pojedynczy znaków.

5. **long** - czyli długi

   Typ przeznaczony do przeznaczona do przechowywania dużych liczb całkowitych za pomocą ośmiu bajtów pamięci.

6. **float** - czyli floating point

   Typ przeznaczony do przechowywania wartości zmiennoprzecinkowej (z częścią ułamkową) za pomocą czterech bajtów.

7. **short** - czyli krótki

   Typ przeznaczony do przechowywania liczb całkowitych za pomocą dwóch bitów.

8. **byte** - czyli bajt albo osiem bitów

   Typ przeznaczony do przechowywania wartości za pomocą jednego słowa komputerowego w zakresie od -128 do +127.

Zakres zmiennych

W pamięci komputera nie jest możliwe zarówno przechowywanie liczb o dowolnej wielkości, jak i z nieskończoną dokładnością. Przyczyną tego jest problemu jest ograniczona ilość pamięci dostępnej w komputerze. W Javie dostępne są metody reprezentowania bardzo dużych liczb zajmujących całą dostępną pamięć, ale nawet w takim przypadku kiedyś ona się kończy.

Najprościej można to wyjaśnić na zmiennej typu byte,której praktycznie nie wykorzystujemy w czasie pisania programów, właśnie ze względu na mały zakres reprezentowanych wartości.

Zmienna typu byte zajmuje osiem bitów w pamięci komputera. Te osiem bitów jest najmniejszym obszarem adresacji, czyli najmniejszym obszarem którym komputer może odczytać lub zapisać.

Osiem bitów z których każdy może występować w jednym z dwóch stanów — jeden albo

zero — daje 2^8 czyli 256 kombinacji czyli możliwość reprezentacji 256 liczb. W Javie przyjęto, że będą to liczby od -128 do +127.

A co się stanie jak przekroczymy ten zakres? Zasadniczo nic. Po prostu liczby zpoza tego zakresu nie są reprezentowane.

```
 127 + 1 = -128

-128 - 1 =  127
```

Podobnie w przypadku zmiennej typu short używającej dwóch bajtów istnieje 2^{16} kombinacji, co daje możliwość reprezentacji 65 656 liczb. W Javie przyjęto, że są to liczby z zakresu od -32 768 do + 32 767.

Natomiast zmienna typu int, którą będziemy wykorzystywać bardzo często wykorzystuje 4 bajty do reprezentacji liczb całkowitych, co odpowiada w Javie zakresowi od - 2 147 483 648 do +2 147 483 647.

Zmienne przeznaczone do reprezentowania liczb z częścią ułamkową mają także swoje ograniczenia reprezentacji. Dotyczy do zarówno zakresu reprezentowanych wartości jak i precyzji z jaką są reprezentowane.

Zasadniczo typy zmiennoprzecinkowe nie reprezentują wartości dokładnie.

Przykładowo typ double z którego będziemy korzystać wykorzystuje osiem bajtów, czyli 264 kombinacji co w rezultacie daje możliwość reprezentacji liczb w zakresie od +/- $1{,}79769313486231570 * 10^{308}$. Są to ogromne wartości, które jednak także mają swoje wartości graniczne.

Deklarowanie zmiennych

Aby zapisać jakąś wartość w pamięci komputera należy najpierw zarezerwować obszar pamięci o określonej wielkości tak, aby uniemożliwić innym programom możliwość jego modyfikacji i w rezultacie zapobiec utracie danych. Dokonujemy tego poprzez deklarowanie zmiennych.

Zmienne typów podstawowych w Javie mogą występować w różnych kontekstach (zmienne lokalne metod, zmienne instancji obiektów, zmienne statyczne klas), a pamięć dla nich może być alokowana i zwalniania w różnych momentach działania programu, tak czy inaczej musi być ona zarezerwowana przed użyciem danej zmiennej i pozostać zarezerwowana tak długo jak jest potrzebna.

Zmienne typów wymienionych powyżej deklarujemy podając typ zmiennej i nazwę zmiennej.

Przykładowo aby zadeklarować zmienną do przechowywania jednej liczby całkowitej należy

napisać:

```
int liczba;
```

Większą ilość zmiennych można deklarować oddzielnie w kolejnych liczbach albo jedna po drugiej oddzielone przecinkami.

```
int liczba1, liczba2, liczba3;
```

Zmienne innych typów podstawowych deklaruje się w identyczny sposób.

String

String jest typem zmiennej przeznaczonym do przechowywania łańcuchów tekstu. Nie należy do typów podstawowych wymienionych wcześniej i jego nazwa pisana jest wielką literą.

Zmienne typu String deklaruje się podobnie jak zmienne typów podstawowych.

```
String adres;
```

Inicjacja zmiennych

Zmienne w Javie mają wartości domyślne i w związku z tym zwykle nie ma konieczności przypisywania im wartości ani zaraz po zadeklarowaniu, a także przed ich użyciem. Jednak często środowiska programistyczne wymuszają takie przypisania np. w przypadku gdy program zawiera instrukcje warunkowe czy pętle.

Inicjacji dokonujemy przypisując wartość do zmiennej.

```
liczba = 5;

cena = 10.5;

znak = 'x';

kobieta = false;

adres = "Marszałkowska 1";
```

W przypadku zmiennej typu String podanej w ostatnim przykładzie nie jest to jedyny sposób inicjacji, ale nie jest to tematem tej instrukcji.

Znak "równości"

```
liczba = 5;
```

Przedstawiony wcześniej zapis przypomina zapis matematyczny, ale nim nie jest.

Po pierwsze znak = nie jest znakiem równości, czyli symbolem porównania dwóch wartości po lewej i prawej stronie.

Po drugie symbol po lewej stronie wcale nie jest zmienną, czyli rodzajem kontenera do przechowywania wartości, lecz de facto jest adresem miejsca w pamięci komputera.

Po trzecie wyrażenie po prawej stronie zasadniczo wcale nie jest wartością liczbową, ale tzw. literałem ją reprezentującym.

```
liczba4 = 5L;
```

W powyższym przypadku zapis 5L oznacza wartość 5 w reprezentacji 64-bitowej, różnej od standardowej reprezentacji 32-bitowej (typ int).

Poniższy zapis z matematycznego punktu widzenia jest bezsensowny.

```
liczba = liczba + 5;
```

Natomiast z punktu widzenia informatyki nie tylko jest uprawniony, ale klarowny i oczywisty.

Natomiast dozwolony w matematyce poniższy zapis jest być błędny w programowaniu.

```
liczba + 5 = 10;
```

Dlatego też mówiąc o znaku równości należy mieć na myśli przypisywanie wartości a nie ich porównywanie.

Porównywanie wartości jest tematem innej instrukcji dotyczącej instrukcji warunkowej [1].

Dowiedz się więcej

1. **Memory Allocation of Primitive, Non-primitive Data Types**
   www.scientecheasy.com/2020/06/memory-allocation-primitive-nonprimitive.html

2. **Variables in Java | Types of Variables**
   www.scientecheasy.com/2020/05/variables-in-java.html

3. **Instrukcja warunkowa**
   https://docs.google.com/document/d/1WT9FxoZbU7BNZj6tOXSx4LubrmkrmSRfGOHpbR4il3M/