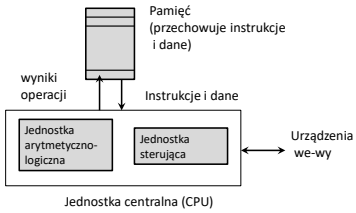
Architektura von Neumanna



Powtarzaj:
Pobierz instrukcję wskazaną przez licznik programu
Ustaw licznik na kolejną instrukcję
Dekoduj instrukcję
Wykonaj instrukcję
end

Dlatego centralnym pojęciem języków imperatywnych jest zmienna jako abstrakcja komórki pamięci

1

Implementacja języków

Programy w dowolnym języku, w dowolnym paradygmacie są finalnie wykonywane na komputerach o architekturze von Neumanna.

Kompilacja : analiza leksykalna → analiza składniowa → kod wstępny → optymalizacja kodu → kod maszynowy – szybkie wykonanie ale …
Interpretacja: kod realizowany instrukcja po instrukcji – wolne wykonanie ale ..
Implementacje hybrydowe (np. java, .net): kompilacja do kodu pośredniego → interpretacja instrukcji kodu pośredniego na maszynie wirtualnej

Programy mają strukturę modularną, kod programisty korzysta z różnego rodzaju bibliotek (C, Fortran), gotowych klas (Java). Po kompilacji następuje więc często **konsolidacja** (linkage). Ostatecznie pewne elementy finalnego programu mogą być określone już na etapie kompilacji, inne dopiero na etapie wykonania.

Znane Wam przykłady języków: kompilowanych, interpretowanych, realizowanych hybrydowo ???

2

Jak opisujemy języki programowania

- **Składnia**:
  - jakie symbole są dopuszczalne
  - jak zapisuje się wyrażenia
  - jak zapisuje się instrukcje sterujące (jakie są w języku) przykłady??
  - Jak (czy?) zapisuje się deklaracje
- Składnię dość łatwo jest opisać formalnie używając np. notacji BNF (Backus-Naur Form)

Np.. typ ::= char | int | float | double // | znaczy LUB
instr_warunkowa ::= if wyr_logiczne then instr [else instr] // [elt opcjonalny
lista_arg ::= arg { "," arg } // { element może być wielokrotnie powtórzony

- **Semantyka**: znaczenie przypisywane symbolom i zapisom operacji – tu trudno o formalne metody opisu

3

Atrybuty zmiennej

Zmienna jest abstrakcją komórki pamięci
Cechy opisujące zmienną:
- nazwa
- adres
- wartość
- typ - zbiór dopuszczalnych wartości, wiąże się z tym zbiór dopuszczalnych operacji
- okres życia – czas od alokacji do usunięcia z pamięci
- zakres widoczności – zmienne występują w podprogramach i są lub nie są widoczne/dostępne w innych podprogramach

4

Atrybuty zmiennej

Nazwa – długość nazwy, dopuszczalne znaki, rozróżnialność wielkich/małych liter, konwencje nazewnicze (zmienneJava, 8 znaków, litera na początku, I – N integer, reszta real), użycie znaków specjalnych - $zm,

Adres – adres 'fizycznego' miejsca w pamięci
to nie musi być zawsze dokładnie ten sam adres (fizyczny), nie ma jednoznacznego przypisania
w programie mogą występować zmienne o tej samej nazwie i różnych adresach w pamięci
Czasem do tej samej komórki pamięci dociera się za pomocą różnych nazw np. int x, *p, *q; p = &x; q = &x; (aliasowanie, zwykle niebezpieczne)
Wartość – kiedyś liczby i znaki, teksty? Wielowartość?(tablice) Obiekty

5

Wiązania

- Różne atrybuty zmiennej można określić (powiązać ze zmienną , *ang. bind*) w różnych momentach jej czasu życia
  - podczas kompilacji (np. powiązanie zmiennej z typem)
  - w czasie ładowania programu do pamięci (powiązanie zmiennej z adresem pamięci)
  - w czasie konsolidacji (wywołanie procedury bibliotecznej wiąże się z kodem tej procedury)
  - w czasie wykonania zmienna (lokalna) jest powiązana z adresem na stosie
- Wiązania statyczne – realizowane przed wykonaniem programu i pozostają niezmienne przez cały czas jego działania
- Wiązania dynamiczne – realizowane w czasie wykonania programu

6

Przykład

```
int x;
….
x = x +6;
```

Wiązanie zmiennej x z typem int odbywa się w czasie kompilacji
Wiązanie typu int z jego zakresem wartości odbywa się jeszcze w czasie projektowania języka
Wiązanie symbolu * z operacją mnożenia – podczas kompilacji
Wiązanie zmiennej x z wartością – podczas wykonania

7

Wiązanie typu ze zmienną

Jak się określa typ zmiennej

- Jawna deklaracja (integer N; String nazwa;)
- Deklaracja niejawna (np. Fortran, Perl wynika z przyjętych konwencji nazewniczych)
- Brak deklaracji - php, Basic, Ruby, Perl
- Wnioskowanie o typie z kontekstu – php, języki funkcyjne
  np. na podstawie wyrażenia x = z * 3.14 wnioskujemy, że x i z są float, a dla f(x) = 2*x zakłada się x integer

Kiedy następuje wiązanie zmiennej z typem?

- Dla zmiennych deklarowanych - **statycznie** – podczas kompilacji
- W językach bez deklaracji – dynamicznie, przy pierwszym nadaniu wartości, przy czym zmienna może zmieniać typ w cyklu życia

  np. JavaScript: tablica = [1.3, 2.7] … tablica = 3.5

8

Dlaczego wiązania dynamiczne (typu ze zmienną) są kosztowne

- Ustalanie typu odbywa się w trakcie wykonania programu – narzut czasowy
- Ze zmienną trzeba przechowywać jej typ – narzut pamięciowy
- Różne operacje mogą wymagać różnych przydziałów pamięci

- Dynamiczne wiązanie typów zmiennych występuje zwykle w językach interpretowanych

9

Wiązanie pamięci

- Alokacja - przydział określonej komórki (komórek) pamięci zmiennej
- Dealokacja – zwolnienie pamięci przydzielonej zmiennej
- Okres życia zmiennej (w pamięci) – czas między alokacją a dealokacją

Rodzaje zmiennych ze względu na czas życia

- Statyczne – pamięć alokowana na początku programu pozostaje niezmienna do jego końca.
- Dynamiczne na stosie - dotyczy zmiennych lokalnych (występujących w podprogramach)
- Dynamiczne na stercie (ang.heap, kopiec), jawne
- Dynamiczne na stercie, niejawne.

10

Zmienne statyczne

- Przydział pamięci następuje na początku programu i pozostaje do końca
- Główna wada – niemożność obsługi np. wywołań rekurencyjnych
- Zwykle dotyczy zmiennych globalnych i jawnie deklarowanych jako statyczne (C)
- W C, C++, Javie zmienne deklarowane z określeniem static wewnątrz definicji klasy, są stałe (wspólne) jedynie dla tej klasy

11

Zmienne dynamiczne na stosie

- Przydział pamięci następuje gdy wykonanie programu dociera do deklaracji zmiennej
- Przydział dokonywany jest na stosie
- Przydzielona pamięć jest zwalniana z końcem bloku zawierającego tę zmienną
- Zalety: mogą być używane w wywołaniach rekurencyjnych, ogólnie pozwalają oszczędnie gospodarować pamięcią
- Wady: narzuty czasowe związane z alokacją i dealokacją, brak możliwości pamiętania historii.
- Tak są alokowane zmienne w funkcjach języka C i w metodach Javy

12

Zmienne dydamiczne na stercie alokowane jawnie

- Alokowane jawnie przez programistę np. polecenie malloc w C lub new (Java, C, C++)
- Dealokowane jawnie (w C i C++ za pomocą free i delete) lub niejawnie (Java, C# odśmiecanie ang. garbage collection)
- Nie mają nazwy; dostępne są poprzez wskaźnik lub referencję (alokowane/dealokowane przez new/delete).
- Zalety: Mogą być używane do tworzenia dynamicznych struktur danych, np. list wiązanych i drzew.
- Wady: Niska efektywność z powodu pośredniego trybu adresowania i skomplikowanego zarządzania stertą. Także duże ryzyko nadużyć ze strony nieostrożnego programisty.
- Przykład: W poniższej sytuacji wiązanie typu jest statyczne; natomiast wiązanie pamięci jest dynamiczne.
  int *p; p = new int; ... delete p;

13

Zmienne dynamiczne na stercie, alokowane niejawne

- Alokowane i dealokowane niejawnie w trakcie wykonania programu w chwili wykonania podstawienia.
- Przykład: Napisy i tablice w Perlu, JavaScript, php
- Zalety: znakomita elastyczność
- Wady: Wysoki koszt, związany z dynamicznym przechowywaniem atrybutów. Trudne wykrywanie błędów.

14

Zakres widoczności zmiennych

- Zbiór instrukcji programu, dla których zmienna jest widoczna (można się do niej odwołać) , pojęcie dotyczy zazwyczaj zmiennych lokalnych
- Zmienne globalne są widoczne dla wszystkich instrukcji (jednak nie należy ich nadużywać – problemy z pamięcią)
- Zmienna lokalna – zadeklarowana i widoczna w danym bloku (podprogramie)
- Zmienna nielokalna może być widoczna lub niewidoczna w bloku, w którym nie jest zadeklarowana
- Zmienne z bloku nadrzędnego mogą być przesłaniane w bloku podrzędnym
- Zasady rozstrzygania zakresu w danym języku mówią, w jaki sposób odwołania do nazw są wiązane ze zmiennymi.

15

Zakres (widoczności) statyczny

- Rozstrzyganie o zakresie jest oparte na kodzie programu. Występuje w językach wywodzących się z Algolu.
- Dwie kategorie języków z zakresami statycznymi:
  - Dopuszczające zagnieżdżenia podprogramów (Algol, Ada)
  - Niedopuszczające zagnieżdżeń – języki wywiedzione z C
- Odwołanie do zmiennych
  - Napotkawszy odwołanie do zmiennej, kompilator szuka jej deklaracji w najbliższym bloku, następnie w najbliższym bloku, w którym bieżący jest zanurzony (poprzednik statyczny), .. Itd. aż do zakresu globalnego.
  - Zmienne mogą się przesłaniać, wówczas obowiązuje deklaracja z bieżącego bloku

16

Przykład – statyczne zakresy widoczności

```
Function outer() {
  function inner1() {
    var x=13;
    inner2();
  }
  function inner2() {
     var y = x;
     print y;
  }
 var x = 3;
 inner1();
}
```

Jaka wartość zostanie wydrukowana?

Odp: 3
Bo statycznym poprzednikiem inner2 jest outer
W podstawieniu y=x zostanie wzięta wartość x z outer

17

Zakres (widoczności) dynamiczny

Zakres dynamiczny opiera się na kolejności wywołań podprogramów. Kryterium rozstrzygania o zakresie jest więc bliskość czasowa, nie przestrzenna;

Jak się realizują odwołania?

- Napotkawszy odwołanie do zmiennej kompilator odnajduje jej atrybuty w bieżącym bloku
- Jeśli ich tam nie ma, szuka się ich w poprzedniku dynamicznym – bloku, z którego bieżący został wywołany i ew. dalej w kolejnego rzędu poprzedniku dynamicznym.

Jaki teraz będzie wynik programu z poprzedniej strony? 13

Zakresy dynamiczne są stosowana np. w perlu, większość języków stosuje jednak zakresy statyczne

18

Wady zakresów dynamicznych

- Gorsza efektywność, gdyż rozstrzyganie o zakresie musi być wykonywane dynamicznie.
- Nie da się statycznie sprawdzić zgodności typów dla zmiennych nielokalnych.
- Słaba czytelność odwołań.
- Podprogramy są wykonywane w środowisku wcześniej wywołanych podprogramów, które jeszcze nie zakończyły działania. Stąd pomiędzy rozpoczęciem a zakończeniem działania podprogramu jego lokalne zmienne są widoczne dla innych podprogramów, niezależnie od ich bliskości przestrzennej.
- A zatem ponownie problemem jest niekiedy zbyt swobodny dostęp.

19

Okres życia zmiennej

- Zakres widoczności zmiennej wynika z rozmieszczenia kodu, albo kolejności wywołań.
- Okres życia zmiennej to czas od alokacji pamięci dla zmiennej do jej dealokacji
- Np. zmienna zadeklarowaną w metodzie, która nie wywołuje innych metod – zakres widoczności od deklaracji do końca metody, okres życia – podobnie do końca metody.
- Zmienna zadeklarowana jako statyczna w funkcji C, Javy ma zakres widoczności tylko wewnątrz tej funkcji. Czas życia jednak do końca programu.

20

Typy danych

- Zbiór wartości dla zmiennych.
  - Typy proste: integer, double, text
  - Typy złożone: tablice, rekordy/struktury (zbiór wartości dla zmiennej typu rekord?)
  - Podtypy, typy wyliczeniowe
  - Typy abstrakcyjne
- Zbiór operacji wykonalnych na danych określonego typu: dodawanie, mnożenie, porównywanie, podstawianie
- Dla typów prostych w danym języku określone są wielkości przydziału pamięci

- **Typ abstrakcyjny** to konstrukcja języka programowania, w której definiujemy typ (w dotychczasowym rozumieniu) oraz operacje na nim w taki sposób, że inne byty w programie nie mogą operować na danych inaczej niż za pomocą operacji przez nas zdefiniowanych.

21

Implementacja typów danych

- Typy liczbowe – prosta implementacja
- typy napisowe – różne implementacje, java jak typ prosty (klasa String), C – tablice znaków , problemy – szeroki zakres zmienności długości napisu
- Tablice – problem nieokreślonym rozmiarem i właściwą alokacją pamięci, zakresy dla indeksów (np. porządkowe, asocjacyjne), tablice jedno i wielowymiarowe, jakie operacje są wykonalne np. podstawianie?
- Rekordy – implementacja w zasadzie prosta (chyba, że …), dopuszczalne operacje,
- Typy wskaźnikowe – wskazują adresy w pamięci, wskaźnik pusty – null, nil, operacje na wskaźnikach?, „wiszące" wskaźniki, używane do alokowania/dealokowania zmiennych na stercie. Czy są jawne wskaźniki w Javie? Używane podczas alokacji pamięci dla obiektów jawnie (C++), wycieki pamięci bądź niejawnie (Java), garbage collection,

22

Kontrola zgodności typów

(1) Int n; float x, y; x = n + y;
(2) Int n, m, z; z = n/m;

Sprawdzanie, czy w danej operacji typy są zgodne z rodzajem operacji,
Stosowane są niejawne konwersje, o ile operacja jest wykonalna
Przy statycznym wiązaniu typów zgodność może być sprawdzone na etapie kompilacji (i dokonane odpowiednie poprawki?)

Język nazywa się **silnie typowanym**, gdy wszystkie niezgodności typów są restrykcyjnie wykrywane. Języki bez typowania ?

23

Co to znaczy zgodność typów

- Zgodność nazwy – jeśli zmienne zostały zadeklarowane z dokładnie tą samą nazwą typu np. float x, y, z;
- Czy wówczas int x; i longint x; są zgodne?
- Kiedy dwie zmienne typu record są zgodnego typu
- Kiedy dwa obiekty są zgodnego typu

24