

Ada-95

Dariusz Wawrzyniak

Część I

Wskaźniki

Plan

- 1 Typy wskaźnikowe i obiekty wskazywane
- 2 Dynamiczna alokacja pamięci
- 3 Wskaźniki ogólne
- 4 Dostęp do obiektów wskazywanych

Plan

- 1 Typy wskaźnikowe i obiekty wskazywane
- 2 Dynamiczna alokacja pamięci
- 3 Wskaźniki ogólne
- 4 Dostęp do obiektów wskazywanych

Wskaźniki w języku Ada

- Wskaźnik w języku Ada jest odpowiednikiem referencji w innych językach programowania.
- Wskaźnika nie należy utożsamiać z adresem w pamięci.
- Rodzaje wskaźników:
 - wskaźniki na dynamicznie przydzielone obszary pamięci,
 - wskaźniki na obiekty utworzone statycznie,
 - wskaźniki na podprogramy.

Typy wskaźnikowe

- typ wskaźnikowy ograniczony — wskaźniki na obiekty tworzone dynamicznie,
- typ wskaźnikowy ogólny:
 - typ wskaźnikowy na zmienne i stałe z modyfikatorem **constant** — wskaźniki na obiekty statyczne lub dynamiczne, za pośrednictwem których można tylko odczytywać wartości wskazywanych obiektów,
 - typ wskaźnikowy na zmienne z modyfikatorem **all** — wskaźniki na obiekty statyczne lub dynamiczne, za pośrednictwem których można zarówno odczytywać wartości wskazywanych obiektów, jak i je zapisywać,
 - typ wskaźnikowy na podprogramy — wskaźniki na funkcje lub procedury, za pośrednictwem których można wywoływać wskazywane podprogramy.

Przykłady definicji typu wskaźnikowego

Przykład: typ wskaźnikowy ograniczony

```
type Pool_Pointer is access Integer;
```

Przykład: typ wskaźnikowy ogólny

```
type RO_Pointer is access constant Integer;  
type RW_Pointer is access all Integer;
```

Przykład: typ wskaźnikowy na podprogram

```
type F_Pointer is access  
    function(a: Float) return Float;  
type P_Pointer is access  
    procedure(a: in out Float);
```

Plan

- 1 Typy wskaźnikowe i obiekty wskazywane
- 2 Dynamiczna alokacja pamięci**
- 3 Wskaźniki ogólne
- 4 Dostęp do obiektów wskazywanych

Tworzenie obiektów dynamicznych

- Pamięć alokowana dynamicznie przydzielana jest na potrzeby obiektów określonego typu.
- Alokator obiektu dynamicznego składa się z instrukcji **new** wraz z typem obiektu i ewentualnym inicjalizatorem, np.: **new Integer'(0)**.
- Alokator zwraca wskaźnik na dynamicznie utworzony obiekt. Wskaźnik ten może być podstawiony pod zmienną wskaźnikową dowolnego rodzaju, zgodną co do typu podstawowego (wskazywanego) z typem alokowanego obiektu.

Przykłady tworzenia obiektów dynamicznych

Przykład tworzenia rekordu

```
type Rec is record  
    i: Integer;  
    f: Float;  
end record;  
type Rec_Ptr is access Rec;  
rp1: Rec_Ptr := new Rec'(i => 1, f => 3.14);  
rp2: Rec_Ptr := new Rec'(1, 3.14);
```

Zwalnianie pamięci przydzielonej dynamicznie

- Automatyczne zwalnianie pamięci przydzielonej do obiektu utworzonego dynamicznie po zlikwidowaniu wszystkich wskaźników do tego obiektu (ang. garbage collection).
- Jawne zwalnianie pamięci poprzez wywołanie specjalnie w tym celu utworzonej procedury. Procedura tworzona jest poprzez konkretyzację wzorca (tzw. procedury rodzajowej, ang. generic) `Unchecked_Deallocation`, dostarczanego jako część środowiska języka Ada.

Przykład wywołania procedury zwalniania

```
Dealloc_Rec( rp2 );
```

Procedura zwalniania

Specyfikacja rodzajowej procedury zwalniania

```
generic  
  type Object (<>) is limited private;  
  type Name is access Object;  
procedure Unchecked_Deallocation  
  (X : in out Name);
```

Przykład konkretyzacji procedury zwalniania

```
with Unchecked_Deallocation;  
procedure Dealloc_Rec is new  
  Unchecked_Deallocation(Rec, Rec_Ptr);
```

Plan

- 1 Typy wskaźnikowe i obiekty wskazywane
- 2 Dynamiczna alokacja pamięci
- 3 Wskaźniki ogólne**
- 4 Dostęp do obiektów wskazywanych

Uzyskanie wskaźnika

Wartość wskaźnika ogólnego uzyskuje się za pomocą atrybutu **access** obiektu wskazywanego.

Przykład

```
gp: RW_Pointer := j'access;  
rp: RO_Pointer := c'access;
```

Uwaga

W celu umożliwienia uzyskania wskaźnika na stałą lub zmienną należy ją zadeklarować jako **aliased**.

Przykład

```
j : aliased Integer;  
c : aliased constant Integer := 87;
```

Wskaźnik na stałą

Wskaźnik na stałą **musi być** typu wskaźnikowego z modyfikatorem **constant**.

Przykład

```
type Const_Pointer is access constant Integer;
```

Uwaga

Jeśli wskaźnik z modyfikatorem **constant** wskazuje na zmienną, może on być używany tylko do odczytu wartości tej zmiennej, nie może natomiast być użyty w celu modyfikacji jej wartości.

Wskaźnik na podprogram

Wskaźnik na podprogram uzyskuje się również za pomocą atrybutu **access**.

Przykład

```
function Func(a: Float) return Float is  
begin  
    ...  
end;  
  
fp: F_Pointer := Func'access;
```


Plan

- 1 Typy wskaźnikowe i obiekty wskazywane
- 2 Dynamiczna alokacja pamięci
- 3 Wskaźniki ogólne
- 4 Dostęp do obiektów wskazywanych

Dostęp do wartości obiektu typu prostego

Dostęp do wartości zmiennej lub stałej typu prostego za pośrednictwem wskaźnika umożliwia pseudoskładowa **all**.

Przykład

```
declare
  gp: RW_Pointer := ...;
  rp: RO_Pointer := ...;
  i: Integer;
begin
  gp.all := 2;
  i := rp.all;
end;
```

Dostęp do składowych rekordu (1)

Dostęp do wartości pól rekordu za pośrednictwem wskaźnika wygląda składniowo się tak samo, jak za pośrednictwem identyfikatora zmiennej (nie jest wymagane użycie **all**).

Przykład

```
declare  
  type Rec is record s: Integer;  
    end record;  
  type Rec_Pointer is access Rec;  
  gp: Rec_Pointer := new Rec;  
begin gp.s := 2;  
end;
```

Dostęp do składowych rekordu (2)

Uwaga

Odniesienie do rekordu jako całości wymaga użycia pseudoskładowej **all**.

Przykład

```
declare
  gp1, gp2: Rec_Pointer;
begin
  gp1 := gp2; -- podstawienie wskaźników
  gp1.all := gp2.all; -- podstawienie wart.
end;
```

Wywoływanie podprogramów przez wskaźniki

Jeśli do podprogramu przekazywane są jakieś parametry aktualne, składnia jego wywołania przez wskaźnik jest taka sama, jak za pośrednictwem nazwy (nie jest wymagane użycie pseudoskładowej **all**). Jeśli natomiast podprogram jest bezparametrowy, wymagane jest użycie pseudoskładowej **all**.

Uwaga

Jeśli do wywoływanego podprogramu z parametrami formalnymi nie są przekazywane żadne parametry aktualne (gdyż np. wejściowe parametry formalne mają wartości domyślne) składnia jego wywołania przez wskaźnik wymaga użycia pseudoskładowej **all**.

Przykład wywoływania podprogramu przez wskaźnik

Przykład

```
declare
  type P_Pointer is access
    procedure(a: Float);
  procedure Proc(a: Float := 0) is
    begin ... end;
  pp: P_Pointer := Proc'access;
begin
  pp(3.14);
  pp.all; -- brak parametrow aktualnych
          -- konieczne jest all
end;
```

Część II

Wyjątki

Wprowadzenie

- Wyjątek jest obiektem typu **exception**.
- Jedynym atrybutem wyjątku **w czystym języku** jest jego nazwa.
- Wyjątki zgłaszane są instrukcją **raise** w programie lub przez środowisko wykonawcze w przypadku napotkania pewnych błędów.
- Po zgłoszeniu wyjątków sterowanie przekazywane jest do strefy obsługi wyjątków.
- Strefa obsługi wyjątków może pojawić się w każdym bloku i jest jego ostatnią częścią.

Deklaracja wyjątków

- Wyjątek, jak każdy inny obiekt, deklarowany jest w części deklaracyjnej bloku.

Przykład

```
x1, x2: exception;
```

- Oprócz wyjątków deklarowanych przez programistę w języku Ada są 4 wyjątki predefiniowane:
 - Constraint_Error,
 - Program_Error,
 - Storage_Error,
 - Tasking_Error.

Zgłaszanie wyjątków

- Zgłoszenie wyjątków następuje w wyniku wykonania instrukcji **raise**.

Przykład

```
raise x2;
```

- Zgłoszenie wyjątku może nastąpić zarówno w wyniku wykonywania ciągu instrukcji, jak również w wyniku opracowania części deklarycyjnej.
- Instrukcja **raise** bez operandu oznacza ponowne zgłoszenie obsługiwanego wyjątku (w bloku „zewnątrznym”) i może się pojawić tylko w strefie obsługi wyjątków.

Obsługa wyjątków

- Strefa obsługi wyjątków znajduje się w ostatniej części bloku i zaczyna się od słowa **exception**.

Przykład

```
begin
    ...
    raise x2;
    ...
exception
    when x1 => raise;
    when x: x2 =>
        Put_line(Exception_Information(x));
end
```