Algorytmy i Struktury Danych

Prof. dr hab. Stanisław Gawiejnowicz
Wydział Biologii UAM
Semestr letni 2022/2023

Plan wykładu nr 4

- Pojęcie procedury i funkcji
- Procedura, a funkcja
- Wywoływanie procedur i funkcji
- Metody przekazywania parametrów
- Przykłady funkcji i procedur

Algorytmy/programy i procedury

- Bardziej złożone algorytmy (programy) wymagają stosowania **procedur**
- Umiejętne stosowanie procedur pozwala traktować powtarzające się fragmenty algorytmów jak „czarne skrzynki”
- Praktycznie wszystkie znane obecnie języki programowania pozwalają na definiowanie procedur

Programowanie proceduralne

- Przed pojawieniem się procedur znane było pojęcie **makra**, **(ko)rutyny** oraz **podprogramu**
- Podprogramy były znane w XIX wieku - maszyna analityczna Babbage'a (1840)
- FORTRAN (Backus, 1954) - pierwszy język programowania z procedurami

John Backus (1924,2007)

- Jeden z pionierów informatyki
- Kierował zespołem pracującym nad pierwszą wersję **FORTRAN**u
- Współpracował z zespołami przygotowującymi Algol 58 oraz Algol 60
- Współautor **notacji Backusa-Naura**, stosowanej w opisach języków formalnych
- Nagrody: n. McDowella (1967), Narodowy Medal Nauki (1975), n. Turinga (1977)

Procedura, a funkcja

- **Procedura** jest ciągiem elementarnych operacji (akcji), a jej **wywołaniu** odpowiada jedna instrukcja
- **Funkcja** jest ciągiem elementarnych operacji (obliczeń) dających w wyniku jedną wartość, jej wywołanie występuje w wyrażeniu
- Procedura (funkcja) może posiadać listę **parametrów** lub być **bezparametrowa**

Parametr/argument procedury/funkcji

- **Parametr procedury** (f.) jest daną formalną, zdefiniowaną w **nagłówku** procedury (f.)
- **Argument procedury** (f.) jest konkretną wielkością typu zgodnego z typem odpowiadającego mu parametru, przekazywaną do procedury (f.) w trakcie jej wywołania

Wykonanie procedury/funkcji

- **Wykonanie** procedury (f.) obejmuje:
  - sprawdzenie zgodności listy argumentów z listą parametrów,
  - przekazanie wartości argumentów do procedury,
  - przekazanie sterowania do pierwszej instrukcji tej procedury (f.),
  - powrót z procedury (f.) i ewentualnie przekazanie wyniku

Wykonanie procedury/funkcji

- Po wykonaniu procedury (f.) sterowanie jest zwykle przekazywane do instrukcji następującej bezpośrednio po linii wywołania procedury (f.).

Parametry procedur/funkcji

- Istotne znaczenie dla procedur mają **parametry** procedur oraz **zmienne lokalne/globalne**
- Kluczową rolę odgrywa wybór **metody przekazywania argumentów**

Metody przekazywania argumentów

- Metody przekazywania argumentów:
  - przez wartość
  - przez zmienną (referencję)
  - przez stałą (nazwę)
- Różnice między tymi metodami polegają na sposobie, w jaki są traktowane argumenty

Przekazywanie a. przez wartość

- Parametr procedury (f.) przekazywany przez wartość zachowuje się jak zmienna lokalna, tzn.:
  - wartość tego parametru jest widziana jedynie w obrębie danej procedury (f.),
  - żadna zmiana jego wartości, dokonana wewnątrz procedury (f.), nie ma wpływu na wartość argumentu na zewnątrz tej procedury (f.)

Przekazywanie argumentów przez zmienną i przez stałą

- Parametr przekazywany przez zmienną jest traktowany jak zmienna:
  - procedura (f.) działa na zmiennej, a nie na kopii – może nawet zmienić jej wartość
- Przekazywanie parametrów przez stałą:
  - podobne do przekazywania parametru przez zmienną - nie ma kopii parametru
  - wartości takiego parametru nie można zmienić wewnątrz procedury (f.)

Procedury i funkcje

Przykłady

Funkcja obliczania potęgi x^n

```
function POTEGA(x,n)
tmp = 1
for i=1 to n do
   tmp = tmp * x
return tmp
```

Funkcja obliczania potęgi x^n

```
function POTEGA(x,n)
tmp = 1
for i=1 to n do
   tmp = tmp * x
POTEGA = tmp
return
```

Funkcja obliczania potęgi x^n

```
function POTEGA-BINARNA(x,n)
k = n
b = 1
c = x
while not (k == 0)
do if (k mod 2 == 0)
    then k = k div 2
         c = c * c
     else k = k - 1
         b = b*c
POTEGA-BINARNA = b
return
```

Funkcja **mod**

```
function mod(a,b)
tmp = 1
while ((tmp+1)*b <= a) do
   tmp = tmp + 1
mod = a – tmp*b
return
```

Sprawdzanie przestępności roku

```
function przestępny(r)
if ((r mod 4 == 0) and (r mod 100 ≠ 0))
   or (r mod 400 == 0)
then
   przestępny = true
else
   przestępny = false
return
```

Wypełnianie tablicy wierszami

```
procedure TABLICA_1(n, A[1..n,1..n])
for i = 1 to n do
    A[1,i] = n-i+1
for i = 2 to n do
    tmp = A[i-1,n]
    for j = n downto 2 do
        A[i,j] = A[i-1,j-1]
    A[i,1] = tmp
return
```

- **UWAGA:** zakładamy, że A jest tablicą globalną!

Wypełnianie tablicy wierszami

- Dla n=4 procedura TABLICA_1 działa następująco:
- 4 3 2 1 // po pierwszej pętli **for**
- tmp=A[1,4]=1, i=2
- 4 3 2 // wypełnianie tablicy od prawej
- 1 4 3 2 // wstawienie tmp
- tmp=[2,4]=2, i=3
- 1 4 3 // wypełnianie tablicy od prawej
- 2 1 4 3 // wstawienie tmp
- tmp=[3,4]=3, i=4
- 2 1 4 // wypełnianie tablicy od prawej
- 3 2 1 4 // wstawienie tmp

Wypełnianie tablicy kolumnami

```
function TABLICA_2(n, A[1..n,1..n])
for i = 1 to n do
    A[i,n] = i
for j = n-1 downto 1 do
    tmp = A[1,j+1]
    for i = 1 to n-1 do
        A[i,j] = A[i+1,j+1]
    A[n,j] = tmp
return
```

Wyszukiwanie w tablicy

```
function ZNAJDZ_1(n,A[1..n+1],x)
A[n+1] = x
i = 1
while (A[i]≠x)
  do i = i+1
if (i<=n)
   then ZNAJDZ_1 = i
   else ZNAJDZ_1 = 0
end
```

Wyszukiwanie binarne w tablicy

```
function ZNAJDZ_2(n,A[1..n],x)
i = 1
j = n
k = 1
while ((A[i]≠x) and (i<=j))
do
      k = (i+j) div 2
      if (x>A[k])
      then i = k+1
      else j = k-1
if (A[k] ≠ x)
   then return 0
   else return k
```