Wstęp do programowania

Procedury i funkcje

Po co procedury i funkcje?

- Gdyby jakis tyran zabronił korzystać z procedur lub funkcji, to informatyka by upadła!
- Procedury i funkcje dostarczają nam niezwykle wygodnego mechanizmu abstrakcji, czyli uwalniania się od nieistotnych szczegółów.
- Są opakowaniem, które ujmuje fragmenty kodu w jedną całość. Przy takim opakowaniu określamy sposób komunikacji tego fragmentu kodu z otoczeniem poprzez parametry.

Algorytm Gaussa obliczania dnia Wielkanocy

Podziel	przez	iloraz	reszta
rok R	19		a
rok R	100	b	c
b	4	d	e
b+8	**25**	f	
b-f+1	3	g	
19a+b-d-g+15	30		h
c	4	j	k
32+2e+2j-h-k	7		s
a+11h+22s	451	u	
h+s-7u+114	31	X	Y

Wielkanoc przypada dnia Y+1 miesiąca X

Procedura Wielkanoc

```
procedure Wielkanoc(R:Integer; var dzien,miesiac:Integer);
{Dla zadanego roku R wyznacza miesiąc i dzień  Wielkanocy}
var a,b,c,d,e,f,g,h,j,k,s,u :Integer;
begin
 a := R mod 19;
 b := R div 100;
 c := R mod 100;
 d := b div 4;
 e := b mod 4;
 f := (b+8) div 25;
 g := (b+f-1) div 3;
 h := (19*a+b-d-g+15) mod 30;
 j := c div 4;
 k := c mod 4;
 s := (32+2*e+2*j-h-k) mod 7;
 u := (a+11*h+22*s) div 451;
 miesiac := (h+s-7*u+114) div 31;
 dzien := (h+s-7*u+114) mod 31+1;
end;
```

Wywołanie procedury

- <instrukcja> ::= <wywołanie procedury>
- <wywołanie procedury> ::=
  <identyfikator procedury><parametry>;
- <identyfikator procedury>::=<identyfikator>
- <parametry>::=ε | (<lista parametrów>)
- <lista parametrów> ::= ε | <parametr>{,<parametr>}
- <parametr>::= <wyrażenie> | <zmienna>

Dodatkowe ustalenia

- Przy wywołaniu procedury musimy pamiętać o tym, że
  - parametrów powinno być tyle samo ile przy deklarowaniu procedury
  - typy parametrów przy deklaracji (formalnych) i przy wywołaniu (aktualnych) muszą się zgadzać
  - Parametry mogą być wywoływane przez wartość lub przez zmienną. W tym drugim przypadku jedyną dopuszczalną formą wyrażenia przy wywołaniu procedury jest <zmienna> .

Wywoływanie przez wartość i przez zmienną

- Różnica między wywołaniem przez wartość i przez zmienną jest taka, że
  - w przypadku wywołania przez wartość przekazujemy wartość wyrażenia procedurze, która korzysta z tej wartości i nie jest w stanie jej zmienić u tego, który wywołuje procedurę
  - w przypadku wywołania przez zmienną przekazujemy adres zmiennej, na której procedura będzie działać – w szczególności będzie mogła zmienić wartość zmiennej. Jest to jeden ze sposobów przekazania wyliczonej wartości na zewnątrz.
- W przypadku procedury Wielkanoc mieliśmy jeden parametr wywoływany przez wartość (R:Integer) i dwa parametry wywoływane przez zmienną (var X,Y:Integer).

Wywoływanie przez stałą

- Jest kłopot, jesli chcemy wywołać przez wartość dużą strukturę danych, na przykład tablicę, w której zamierzamy namierzyć jakąś wartość. Tablicy zmieniać nie będziemy, więc zgodnie z zasadami sztuki powinniśmy wywołac ten parametr przez wartość, ale to jest kosztowwne
- W tym celu w rozszerzeniach Pascala (nie jest to standard!) stosuje się trzeci sposób: wywołanie przez stałą.
  - Function jest(s:Integer; const A:tablica):Boolean
- Takie wywołanie spowoduje przekazanie adresu tablicy A jako parametru, ale z zakazem zmiany wartości A. Zatem A może występować tylko po prawej stronie instrukcji przypisania.

Przykład

- R:=2011;
- Repeat
  - R:=R+1;
  - Wielkanoc(R,x,y);
- Until (x=3) and (y=22)
  -
- TEN KOD OBLICZY NAM PIERWSZY ROK LICZĄC OD DZIŚ, W KTÓRYM BĘDZIE NAJWCZEŚNIEJSZA MOŻLIWA WIELKANOC.

Implementacja procedur

- Wywołanie procedury (funkcji) powoduje zawieszenie wykonywania bieżącego kodu, zapamiętanie stanu obliczeń oraz przygotowanie środowiska do uruchomienia procedury i to w taki sposób, aby umożliwić powrót do przerwanych obliczeń
- Robi się to za pomocą stosu (systemowego) i rekordów aktywacji, które się na tym stosie odkłada
- Rekord aktywacji zawiera wszystkie informacje konieczne do odtworzenia środowiska wywołania.

Rekord aktywacji zawiera

- Stan procesora w momencie wywołania procedury
- Adres powrotu
- Parametry procedury
  - Parametry wołane przez wartość mają komórki przeznaczone na ich wartość
  - Parametry wołane przez zmienną mają komórki przeznaczone na adresy tych zmiennych
- Zmienne lokalne procedury
- Dowiązanie dynamiczne - adres poprzedniego rekordu aktywacji na stosie
- Dowiązanie statyczne - adres rekordu aktywacji procedury, która poprzedza naszą w drzewie deklaracji i jest najwyżej na stosie

Stos systemowy

- Rekordy aktywacji mają z reguły różną długość
- Nic nie stoi na przeszkodzie, żeby na stosie pojawiły się rekordy aktywacji tej samej procedury (rekursja)!
- Wielkość stosu w czasie wykonania programu wchodzi w skład kosztu pamięciowego algorytmu
- Stos ma ograniczoną wielkość, ustalaną na poziomie parametrów środowiska, stąd możliwość błędu spowodowanego brakiem pamięci, gdy rekordów jest tak dużo, że na kolejny już nie ma miejsca

Widoczność zmiennych

Kiedy w procedurze występuje zmienna, system operacyjny szuka właściwego adresu w pamięci (kontekstu na stosie) - pod tą samą nazwą mogą występować różne obiekty i trzeba znaleźć właściwy.

Zaczynamy od bieżącego rekordu aktywacji i przechodząc po dowiązaniach statycznych dochodzimy do piereszego miejsca, gdzie w rekordzie aktywacji znajduje się komórka związana z tą zmienną. Jest to „najświeższa” wersja tej zmiennej.

Rekord aktywacji programu głównego jest na dnie stosu

Widoczność zmiennych przykład

```
var x,y:Integer;
Procedure A(x:Integer);
   forward C;
   procedure B (var y:Integer); begin Write(x);  y:=x; C end;
   procedure C; begin Write(x); y:=1; if x>0 then x:=0 else b(y) end;
begin {A}
 Write(x);   x:=2;
 if x>y then B(y) else y:=3;
 C
end {A};
begin {program główny}
 x:=0; y:=0; a(y)
end.
```

Funkcje

- Często efektem działania procedury ma być jakiś wynik :)
- W przypadku, gdy wynik jest typu prostego (Integer, Real, Boolean, Char, String, typ wskaźnikowy) możemy użyć funkcji, która jest procedurą z końcowym wynikiem.
- Przykład:

```
Function Mniejsze (x,y:Liczba):Boolean;
var i:Integer; {Liczba=array[0..n]:Integer}
begin
 i:=n;
 while (i>0) and (x[i]=y[i]) do i:=i-1;
Mniejsze:=x[i]<y[i]
end;
```