



# SYNTAX LOGIC CSPRO

Nucke Widowati KP, S.Si, M.Sc

# Pengantar Sintaksis Program CSPro

- Merupakan kendali dan cek operasional pemasukan data
- Ditulis dalam Form designer → View-logic untuk elemen yang berkaitan
- Diletakkan pada seluruh aplikasi, level, formulir, field, roster/group



# Prosedur pada CSPro

## Prosedur global

- Berisi deklarasi dan definisi
- Tidak dapat berisi pernyataan yang dapat dieksekusi (kec. Membuat fungsi sendiri)
- Pada awal file logika 'PROC GLOBAL'



# Prosedur pada CSPro

## Prosedur File Formulir

- Preproc ( pernyataan yang ada dibawahnya akan dieksekusi pada awal blok)
- Postproc ( pernyataan yang ada dibawahnya akan dieksekusi pada akhir blok)



# Prosedur pada CSPro

Prosedur level, formulir, roster dan field

- Berisi pernyataan yang dapat dieksekusi atau penugasan.
- Semua pernyataan akan diasumsikan postproc kecuali jika secara eksplisist dinyatakan preproc



# Perintah Preproc dan Postproc

- Preproc adalah perintah membaca kondisi persyaratan sebelum data di input kedalam variabel yang ditetapkan.
- Postproc adalah perintah membaca kondisi persyaratan setelah data di input kedalam variabel yang sedang dijalankan.





# Contoh:

- penempatan preproc dan postproc pada variabel PROP

```
PROC PROP
```

```
preproc
```

```
$=b1r1;
```

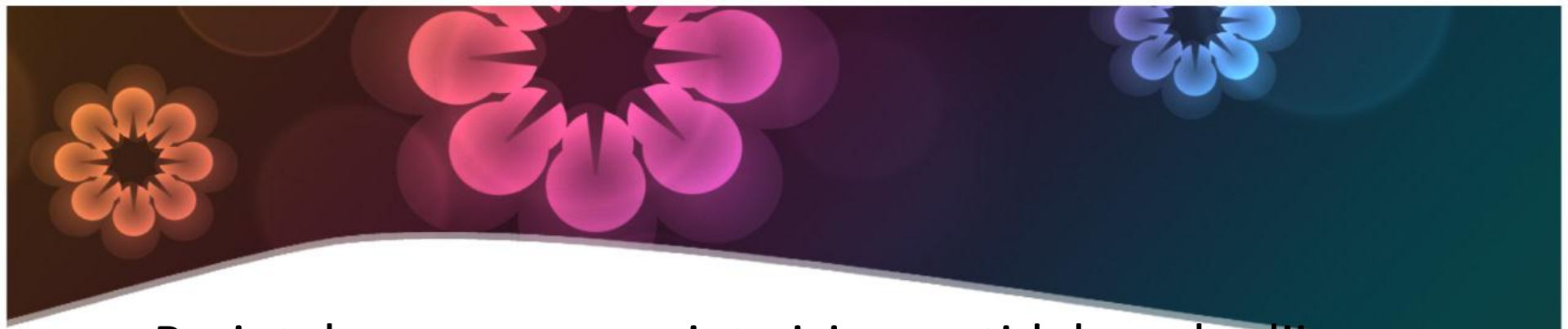
```
noinput;
```

```
postproc
```

```
if (B1R1<>$) then e=err;
```

```
    reenter B1R1;
```

```
endif;
```

- 
- Perintah *preproc* meminta isi prop tidak perlu diinput (*noinput*), nilainya diambil dari nilai-nilai variabel b1r1 yang sudah terlebih dahulu terisi di awal pengisian.
  - Perintah *postproc* memberikan perintah untuk memberikan kepastian saja, apabila isian b1r1 (diawal pengisian blok 1 pertanyaan pertama), tidak sama dengan \$ (prop), maka diberikan instruksi *error* yang artinya proses terhenti sampai dengan petugas membetulkan pengisian yang benar. Perintah *reenter* b1r1 menunjukkan kursor akan kembali ke posisi b1r1 untuk mengisi ulang isian yang betul tersebut





# Perintah *If, Then dan Reenter*

- Perintah if dan then sering digunakan secara berpasangan dalam instruksi untuk proses pengecekan dua atau lebih variabel
- Perintah if dan then biasanya diikuti oleh instruksi error, bisa ditulis dengan e saja, diikuti perintah reenter dan endif

# Contoh:

```
PROC B5R1G  
if $=1 and umur=0 then  
e=display("APA BENAR BAYI  
SAKIT GIGI", umur);  
reenter;endif;
```

- Lambang \$ adalah kata lain dari nama variabel yang ruangnya sedang di tuliskan instruksi program, alias dari \$ adalah b5r1g



# Perintah *If, Then dan Reenter*

- Penulisan selanjutnya *reenter;endif;* dua kata tersebut memberikan instruksi loncatan kebelakang ke arah variabel yang sama (program berhenti sampai dengan persyaratan diperbaiki menjadi benar).
- Penulisan *reenter* kemudian diikuti dengan nama variabel yang berada di awal atau sebelum variabel yang sedang di input diperbolehkan,



# Contoh:

```
reenter b1r1a;
```

- Perintah tersebut memberikan instruksi loncat kembali ke
- belakang sampai dengan posisi input variabel b1r1a.

# Perintah Melompati (Skip to Next)

- Perintah melompati pengisian berdasarkan persyaratan kondisional tertentu sesuai dengan alur pengisian data dengan menggunakan perintah skip to.



## Contoh:

```
PROC B4K8
```

```
preproc
```

```
if b4k5>6 and b4k1<b2r2 then  
skip to next b4k1;endif;
```

```
if b4k5>6 and b4k1=b2r2 then  
endsect;endif;
```

- Perintah diatas memberikan perintah apabila  $b4k5 > 6$  (umur lebih besar dari 6 tahun) dan  $b4k1 < b2r2$  (dan nomor urut art ( $b4k1$ ) masih lebih kecil dari jumlah anggota rumah tangga ( $b2r2$ )), maka selanjutnya loncatan kursor (skip to) mengarah pada  $b4k1$  (next  $b4k1$ ) yaitu anggota rumah tangga selanjutnya.





## Contoh:

- Biasanya digunakan juga perintah pendukung lainnya yaitu *endsect*.
- Penggunaan *endsect* untuk memberikan eksekusi jika persyaratan sudah terpenuhi maka proses input data di section tersebut sudah selesai dan dilanjutkan ke section selanjutnya.

# Perintah Melompati (Skip to Next)

- Penggunaan perintah skip yang sering digunakan adalah melewati variabel yang tidak relevan untuk diinput seperti contoh, yaitu apabila isian b5r7 sama dengan kode 2, maka lompat pengisian langsung ke b5r9a.
- Dengan demikian apabila isian b5r7 berkode 1, maka alur program tidak meloncat ke b5r9a.



## Contoh:

```
PROC B5R7  
if $=2 then skip to b5r9a;  
endif;
```



# Perintah Box

- Perintah box digunakan untuk mengecek pengisian terdahulu dalam bentuk yang lebih kompleks.
- Apabila persyaratan pengecekan melibatkan lebih dari satu variabel dan mempunyai persyaratan yang sangat prinsip harus dilakukan serta kompleks

# Contoh:

Partisipasi  
sekolah

```
PROC B5R15 {tingkat pendidikan}  
box b5r13 : $ : umur =>x;  
2 : 1,2 : 5-17 =>1;  
2 : 3,4 : 11-30 =>1;  
2 : 5,6,7 : 12-64 =>1;  
2 : 8,9,10 : 15-64 =>1;  
2 : 11 : 15-64 =>1;  
3 : 1,2 : 5-98 =>1;  
3 : 3,4 : 11-98 =>1;  
3 : 5,6,7 : 12-98 =>1;  
3 : 8,9,10 : 15-98 =>1;  
3 : 11 : 20-98 =>1;  
: : =>0;  
endbox;  
if x<>1 then  
e=errmsg(064,b5r13,$,umur);  
reenter;  
endif;
```

# Perintah Box

- Program diatas membatasi pengisian sesuai dengan alur logika, dalam box ditetapkan tiga buah variabel yang saling bersinggungan.
- Variabel-variabel tersebut antara lain status pendidikan kode 2 (masih sekolah) dan kode 3 (sudah tidak sekolah lagi), b5r15 (tingkat pendidikan yang ditamatkan mulai dari SD, SMP, SLTA, D3, S1, S2, dan S3), kemudian konsisten dengan umur. Apabila persyaratan yang ditetapkan tidak terpenuhi maka program akan terhenti dan dinyatakan salah.





# Perintah Box

- Untuk yang masih sekolah dengan status SD dan SD sederajat, usianya hanya boleh antara 5 sampai dengan 17 tahun.
- Untuk yang masih sekolah dengan status SMP dan SMP sederajat, usianya hanya boleh antara 11 sampai dengan 30 tahun



# Perintah Box

- Untuk yang masih sekolah dengan status SLTA, SLTA kejuruan, dan SLTA sederajat, usianya hanya boleh antara 12 sampai dengan 64 tahun
- Untuk yang masih sekolah dengan status D1, D3, S1, S2, dan S3, usianya hanya boleh antara 15 sampai dengan 64 tahun
- Untuk yang tidak sekolah lagi dengan status SD dan SD sederajat usianya hanya boleh antara 5 sampai dengan 98 tahun

# Perintah *endif*, *endsect* dan *endlevel*

- adalah perintah-perintah untuk mengakhiri suatu perintah.
- Apabila perintah diawali dengan *if* maka diakhiri dengan *endif*.
- Untuk mengakhiri pengisian satu *section* walaupun belum selesai sampai akhir variabel, apabila harus berakhir maka diberikan perintah *endsect*.



# Perintah endif, endsect dan endlevel

- Sedangkan untuk perintah menyelesaikan satu dokumen apabila belum sampai akhir dokumen akan tetapi harus berakhir sesuai alur pengisiannya maka diberikan instruksi *endlevel*.

# Contoh:

```
PROC B4K6
```

```
postproc
```

```
if $>1 and b4k5<10 then
```

```
e=errmsg(021);reenter;endif;
```

```
if ((B4K3=6 or B4K3=4 or B4K3=2) and $=1)  
then
```

```
e=errmsg(147);reenter B4K3;endif;
```

```
if ((B4K3=2 or b4k3=4) and $<>2) then
```

```
e=errmsg(020);reenter;endif;
```

```
if b4k5>6 and b4k1<b2r2 then skip to next  
b4k1;endif;
```

```
if b4k5>6 and b4k1=b2r2 then endsect;endif;
```

```
PROC B7R4A
```

```
if $=2 then endlevel;endif;
```

# Perintah `pos ( )` then

- Penulisan *pos* adalah perintah posisi *input* data.
- Bisa ditetapkan jenis huruf bukan angka, formasi penulisan perintah tersebut disesuaikan dengan jenis variabel yang dibangun di file *dictionary* dalam format karakter atau numerik.



# Contoh:

```
PROC B1R10A
if !pos($[1:1],alphalst) then
errmsg("NAMA HARUS HURUF,BUKAN ANGKA DAN
HARUS HURUF
BESAR");
reenter;
endif;
```

# Perintah pos ( ) then

- Pertama harus diciptakan terlebih dahulu nama *alphalst* yang merupakan sekumpulan huruf dari a sampai dengan z.
- `alphalst =  
"ABCDEFGHIJKLMNOPQRSTUVWXYZ";`

# Perintah `pos ( )` then

- *!pos* memerintahkan posisi data input harus huruf bukan angka, dan untuk `!pos ($ [1:1] , alphalist)` artinya, pada variabel (\$) yang sedang dalam posisi mau di input posisi pertama harus berupa huruf a sampai dengan z (alphalist).
- Perintah diatas memberikan alias untuk alphalist berupa huruf kapital (huruf besar) dan tidak diperbolehkan berupa angka.



# Perintah Menghitung Jumlah Kasus Tertentu

- Penghitungan jumlah kasus tertentu sebagai kontrol konsistensi pengisian sangat diperlukan.
- Biasanya penempatan penghitungan diletakkan pada section bersangkutan atau setelahnya.

# Contoh:

```
PROC SEC2_ROSTER
```

```
a=count(sec2_ROSTER where b4k5<5);
```

```
if a>b2r21 then e=errmsg(901);reenter b2r21;  
endif;
```

```
b=count(sec2_ROSTER where b4k5>=5 and  
b4k5<=9);
```

```
if b>b2r22 then e=errmsg(902);reenter b2r22;  
endif;
```

```
c=count(sec2_ROSTER where b4k5>=10);
```

```
if c>b2r23 then e=errmsg(903);reenter b2r23;  
endif;
```



# Perintah While dan Do

- Perintah while dan do dituliskan berpasangan, memberikan instruksi membaca atau mengecek jumlah putaran atau pemasukan dari pola input data multiple entry atau yang bersifat berulang dengan memberikan kontrol berdasarkan variabel tertentu yang sudah ditetapkan.



# Contoh:

```
PROC SEC1
```

```
  i = 1 ;
```

```
  while i<=b2R2 do
```

```
    b4k1(i)=i;
```

```
    i=i+1;
```

```
  enddo;
```

# Perintah While dan Do

- Perintah dalam sec1 selanjutnya dimulai dari  $i = 1$ , dapat dinyatakan program akan membaca mulai dari putaran pertama, kemudian penulisan while  $i$  lebih kecil atau sama dengan  $b2r2$  (jumlah anggota rumah tangga) mengisaratkan program akan berjalan apabila terpenuhi kondisi tersebut yaitu nilai  $i$  tidak melebihi nilai  $b2r2$ .

# Perintah While dan Do

- Syarat lainnya diminta apabila  $b4k1(i) = 1$  (nomor urut anggota rumah tangga) datanya bisa dimasukkan mulai dari yang pertama sampai dengan yang terakhir.
- Apabila jumlah  $i > b2r2$  yaitu jumlah anggota rumah tangga sudah melebihi dari yang harus di input, maka program entry berakhir dan secara otomatis berganti ke rumah tangga berikutnya yang harus di input.

# Perintah While dan Do

- Setiap perintah *while.....do* diakhiri dengan *enddo*.
- Format penulisan sebagai berikut *While.....do.....enddo*.
- Untuk memberikan batasan instruksi pertama dengan instruksi ke dua dan berikutnya, maka setiap akhir perintah diberikan tanda semi kolom “;”.

# Perintah Soccurs dan Noccurs (Multiple Entry)

- Perintah melakukan putaran pengisian untuk type data *multiple entry*
- Program untuk memanipulasi putaran *multiple entry* dapat dilakukan dengan memberikan perintah *noccurs* apabila bentuk *entry data multiple* dengan menggunakan perintah *same location* (lokasi yang sama) atau bentuk tabel (*roster*).

# Contoh:

```
PROC B4K1
```

```
preproc
```

```
n=noccurs (sec2_ROSTER) +1;
```

```
    if n<=b2R2 then
```

```
        $(n)=n;
```

```
        noinput;
```

```
endif;
```

```
PROC NART
```

```
preproc
```

```
n=soccurs (SEC05) +1;
```

```
nart (n)=n;
```

# Perintah Soccurs dan Noccurs (Multiple Entry)

- Penggunaan *soccurs* dengan bentuk entry data *multiple* tetapi tampilan pengisian data seolah-olah berbentuk *single*, atau dengan kata lain tampilan entry menggunakan pilihan *different location* (lokasi yang berbeda).
- Perintah tersebut memandu pengisian secara berurutan sehingga jumlah pengisian yang berulang tersebut sesuai dengan persyaratannya.





# Perintah Error dan Display

- *Errmsg* adalah pesan kesalahan yang dipersiapkan untuk membantu operator entry memahami kenapa terjadi kesalahan.
- Dengan opsi ini program akan terhenti dan bisa dilanjutkan kembali proses entry datanya apabila telah diperbaiki sesuai intruksi pesan kesalahan yang ditampilkan tersebut.



# Perintah Error dan Display

- *Display* adalah peringatan kesalahan yang dipersiapkan untuk operator agar lebih hati-hati dalam proses input datanya. Dengan opsi ini program tidak terhenti artinya *proses entry* tetap dapat dilanjutkan hanya saja *operator* mendapat peringatan untuk mengecek kembali isian yang telah dilaksanakannya.

# Perintah Error dan Display

- Model *e=errmsg(140)*, memberikan pesan kesalahan yang paling sederhana di dalam penulisan pesannya.

```
if ($>98 and $<99) then e=errmsg(140);reenter;endif;  
if B4K3=3 and $>=50 then e=display(15,$);endif;  
if B4K3=5 and $>=40 then e=display(15,$);endif;
```

- *Display* yang ditampilkan menampilkan nilai kesalahan (\$)

140 => UMUR TIDAK BOLEH LEBIH DARI 98 TAHUN

015 => UMUR ANAK, CUCU APAKAH BETUL =%D TAHUN?

# Perintah Error dan Display

- Model `e=errmsg(134,b4k5,b4k5(1))` lebih kompleks dimana pesan kesalahan 134 memberikan instruksi `ortu = %d krt=%d` artinya `%d` yang pertama menampilkan isi `b4k5` dan `%d` yang kedua adalah isi dari `b4k5(1)`.

```
if B4K3=6 then if (B4K5-10)<B4K5(1) then  
e=errmsg(134,B4K5,B4K5(1)); endif;endif;  
134 => UMUR ORTU DAN KRT TIDAK WAJAR, ORTU=%D KRT=%
```

# Perintah Error dan Display

- Untuk penulisan `errmsg("HANYA UNTUK UMUR 0-4 TH SAJA");` adalah bentuk penulisan pesan kesalahan lainnya yang bisa dituliskan langsung.

```
if b4k5<=4 and $=missing then errmsg("HANYA  
UNTUK UMUR 0-4 TH SAJA");reenter;endif;  
if b4k5>4 and !$=missing then errmsg("HARUS  
MISSING TEKAN ENTER SAJA");reenter;endif;
```

# Perintah Error dan Display

- Pesan kesalahan menggunakan %d, c dan \$.  
Penggunaan c yaitu menghitung jumlah dari satu variabel (count) yaitu jumlah anak, yang kemudian hasil penjumlahan tersebut ditampilkan dalam pesan kesalahan.

```
if $<>(b5r29b1+b5r29b2) then e=errmsg(091);  
reenter;endif;  
c=count(sec2_ROSTER where b4k3=3);  
if c<>$ and hb=2 then e=errmsg("Jumlah Anak  
di Blok IV = %d, Jumlah Anak Kandung Hidup = %d,Cek  
! ",c,$);endif;
```



# Komentar pada CPro

- Komentar pada CPro
- Membuat alur logika menjadi lebih baik
- Teks komentar tidak akan dieksekusi.
- Dituliskan dalam {}