passed: myfoo not defined
TESTING STARRAY

— properties/sub-structures, one definition at a time —
STtest
passed: dup detected

STtestX
passed: new done!

adding struct ZX to STtest.TT
passed: defined correctly!

STtest.TT Zy=TT Zy-default
passed

STtest.TTT Zy(err)=TT Zy-default
passed: err detected

— Extending def structure of an already instantiated starray —
Note: it will loop if not fixed!
— Fixing it —
Note: STtest def \show shall appear in logs (if fixed)

```
>{STtest} struct =>
>  {X} => {X-default}
>  {Y} => {Y-default}
>  {SS} struct =>
>    {Z} => {SS Z-default}
>    {Zx} => {SS Zx-default}
>    {YY} => {YY-default}
>    {ZsX} struct =>
>      {ZsXa} => {SS ZsX ZsXa-default}
>      {Zsxb} => {SS ZsX ZsXb-default}
>    {TTy} struct =>
>      {TTyZsXa} => {TTyTT-ZsXa-default}
>      {TTyZsxb} => {TTyTT-ZsXb-default}
>      {Z} => {TTy-Z-default}
>  {TT} struct =>
>    {Z} => {TT Z-default}
>    {Zx} => {TT Zx-default}
>    {Zy} => {TT Zy-default}
>    {ZX} struct =>
>      {ZXa} => {TT Zx ZXa-default}
>      {Zxb} => {TT Zx ZXb-default}
```

Note: STtest terms \show shall appear in logs (if fixed)

Note: STtest.SSX (err/warning) shall be in logs:

— testing term_parser function —
STtest.TT.ZXx (err)
passed: err detected

STtest.TT.ZX (correct)
passed

— Texting expandable predicates and command —
passed: myfoo isn't defined
passed: myfoo isn't defined
passed: STtest.TT.ZX is a starray

executing term syntax:n (no output)

This is ZXa:«TT Zx ZXa-default» (using 'parsed' one)

— expandable cnt/iter commands —
The current cnt:2(using 'parsed' one)

passed: cnt isn't 0

The current iter:2(using 'parsed' one)

passed: iter is 2

passed: myfoo not a prop of
passed: ZXa is prop

passed: ZXa is true

— Texting expandable predicates and command with parsed ref variables —
executing term syntax:nNN (no output refA)
executing term syntax:nNN (no output refB)

This is ZXa:«TT Zx ZXa-default» (using 'refA' one)

— expandable cnt/iter commands —
The current cnt:2(using 'refA' one)

passed: cnt isn't 0

The current iter:1(using 'refB' one)

passed: iter is 1

The current iter:2(using 'refA' one)

passed: iter is 2

(using refA):
passed: myfoo not a prop of
(using refA):
passed: ZXa is prop
(using refB):
passed: ZXa wasn't found

passed: ZXa is true

— testing get_prop functions —
Note: This is (default) 'X' property fromSTtest[hah] term:
X-default

Note: Same with a token-list variable
X-default

Note: Same with 'branching'
X:X-default
passed: X found correctly

Xt:

```
passed: Xt don't exist

      Note: (same) testing \...if_in:
passed: X exists
passed: Xty don't exit

  — Testing iter functions —
Current STtest iter:
      Note: direct access:2
      Note: using a tmp var:2
      Note: reseting iter
iter:1
      Note: next iter
iter:2
      Note: next iter
iter:2
      Note: set iter hash
iter:1
      Note: set iter->5
iter:2
      Note: set iter->0
iter:1

  — Testing iter functions with branching —
Current STtest iter:
passed: got: 1
iter from STtestY (err):
passed: syntax err OK
      Note: reseting iter
passed
iter:1
      Note: next iter
passed
iter:2
      Note: next iter
passed: 'saturated'
iter:2
      Note: set iter hash
passed: hash found
iter:1
      Note: set iter->5
passed: 'over'
iter:2
      Note: set iter->0
passed: 'under'
iter:1
      Note: set iter->2
passed
iter:2

  — Testing cnt functions —
Current STtest cnt:
      Note: direct access:2
      Note: using a tmp var:2
  — Testing cnt function with branching —
Current STtest cnt:
passed: got: 2
Current STtestX cnt:
passed: got: 0
Current STtestY cnt:
passed: non existant
```

— Testing _if_in function —
passed: X found
passed: G not found
— Testing _term_parser function —
STtest[2].TT is:
passed: correct
STtest[1].TT is:
passed: wasn't instantiated
STtest[1].GG is:
passed: not correct

— Testing (g)set_prop functions —
STtest[2].TT.Z current value: TT Z-default
STtest[2].TT.Z new value: newZ value

— Testing (g)set_prop inside a group —
STtest[2].TT.Z inside: newZ inside group
STtest[2].TT.Z ouside: newZ value
STtest[2].TT.Z inside gset: newZ gset inside group
STtest[2].TT.Z ouside: newZ gset inside group

— Testing (g)set_prop functions —
STtest[2].TT.Z current value: newZ gset inside group
STtest[2].TT.Z new value: newZ value

— Testing (g)set_prop inside a group —
STtest[2].TT.Z inside: newZ inside group
STtest[2].TT.Z ouside: newZ value
STtest[2].TT.Z inside gset: newZ gset inside group
STtest[2].TT.Z ouside: newZ gset inside group

— Testing (g)set_prop functions with branching —
STtest[2].TT.Z current value: newZ gset inside group
passed: new value: newZZZZ value

— Testing (g)set_prop inside a group —
passed: new value: newZ inside group
STtest[2].TT.Z ouside: newZZZZ value
passed: new value: newZ gset inside group
STtest[2].TT.Z ouside: newZ gset inside group
setting:STtest[1].TT (err, not instantiated)
passed: correct, no instance

— set_prop:nnV inserting a sequence as a property —
Note: the 2 (equal) sequences shall be in log (\show)

— defining/setting from keyval —
— setting from keyval with branching —
passed: correct
— setting from keyval with branching II —
passed: correct
student definition:
>{student} struct =>
> {first} => {–first–}
> {last} => {–last–}
> {name} => {–full-name–}
> {article} => {o(a)}
> {narticle} => {(a)}
> {Article} => {O(A)}
> {Narticle} => {(A)}

```
>   {Nproc} => {—}
>   {ID} => {—}
>   {email} => {—}
>   {advisor} struct =>
>     {first} => {–first–}
>     {last} => {–last–}
>     {name} => {–full-name–}
>     {article} => {o(a)}
>     {narticle} => {(a)}
>     {Article} => {O(A)}
>     {Narticle} => {(A)}
>     {institution} => {–inst–}
>     {titleinfo} => {–info–}
>     {email} => {—}
>     {phone} => {—}
>     {somedata} struct =>
>       {fieldA} => {field-Ax}
>       {fieldB} => {field-B}
>       {fieldC} => {field-C}
>       {fieldD} => {field-D}
>   {reviewers} struct =>
>     {first} => {–first–}
>     {last} => {–last–}
>     {name} => {–full-name–}
>     {article} => {o(a)}
>     {narticle} => {(a)}
>     {Article} => {O(A)}
>     {Narticle} => {(A)}
>     {institution} => {–inst–}
>     {titleinfo} => {–info–}
>     {email} => {—}
>     {phone} => {—}
```

student current terms:

```
>{student[1]} (idx: A) =>
>   {first} => {first name}
>   {last} => {last name}
>   {name} => {–full-name–}
>   {article} => {o(a)}
>   {narticle} => {(a)}
>   {Article} => {O(A)}
>   {Narticle} => {(A)}
>   {Nproc} => {—}
>   {ID} => {—}
>   {email} => {}
>   {advisor[1]} (idx: A) =>
>     {first} => {advisorA first name}
>     {last} => {advisorA last name}
>     {name} => {–full-name–}
>     {article} => {o(a)}
>     {narticle} => {(a)}
>     {Article} => {O(A)}
>     {Narticle} => {(A)}
>     {institution} => {–inst–}
>     {titleinfo} => {–info–}
>     {email} => {—}
>     {phone} => {—}
>     {somedata[1]} (idx: A) =>
>       {fieldA} => {field-Ax}
>       {fieldB} => {field-B}
```

> {fieldC} => {field-C}
> {fieldD} => {field-D}
> {advisor[AB]} (idx: A) =>
> {first} => {advisorA first name}
> {last} => {advisorA last name}
> {name} => {–full-name–}
> {article} => {o(a)}
> {narticle} => {(a)}
> {Article} => {O(A)}
> {Narticle} => {(A)}
> {institution} => {–inst–}
> {titleinfo} => {–info–}
> {email} => {—}
> {phone} => {—}
> {somedata[1]} (idx: A) =>
> {fieldA} => {field-Ax}
> {fieldB} => {field-B}
> {fieldC} => {field-C}
> {fieldD} => {field-D}
> {advisor[2]} (idx: B) =>
> {first} => {student2set first name}
> {last} => {advisorB last name}
> {name} => {–full-name–}
> {article} => {o(a)}
> {narticle} => {(a)}
> {Article} => {O(A)}
> {Narticle} => {(A)}
> {institution} => {–inst–}
> {titleinfo} => {–info–}
> {email} => {—}
> {phone} => {—}
> {somedata[1]} (idx: A) =>
> {fieldA} => {field-Ax}
> {fieldB} => {field-B}
> {fieldC} => {field-C}
> {fieldD} => {field-D}
> {somedata[2]} (idx: B) =>
> {fieldA} => {field-Ax}
> {fieldB} => {field-B}
> {fieldC} => {field-C}
> {fieldD} => {field-D}
> {reviewers[1]} (idx: A) =>
> {first} => {reviewerI first name}
> {last} => {reviewerI last name}
> {name} => {–full-name–}
> {article} => {o(a)}
> {narticle} => {(a)}
> {Article} => {O(A)}
> {Narticle} => {(A)}
> {institution} => {–inst–}
> {titleinfo} => {–info–}
> {email} => {—}
> {phone} => {—}