MiniASN - MiniTutorial

Typy

Dostępne typy proste

UINT | UINT_X

- Liczba całkowita nieujemna. Możliwość parametryzacji rozmiaru w bitach, liczbą całkowitą nieujemną

BITSTRING | BITSTRING_X

- Ciąg bitów. Możliwość parametryzacji rozmiaru w bitach liczbą całkowitą nieujemną

BOOL

- Wartość boolowska TRUE / FALSE(1 bit)

Dostępne typy złożone

```
ARRAY[X]
{
id TYPE
id2 TYPE2
...
}
```

- Tablica. Parametryzacja ilości rekordów w tablicy liczbą całkowitą nieujemną lub zmienną reprezentującą liczbę. Może zawierać jedno lub więcej pól. Pola mogą być typów wcześniej zadeklarowanych, także mogą być parametryzowane

```
CHOICE[X1 X2 ...]
{
TYPE1(condition1)
TYPE2(condition2)
...
TYPEN(DEFAULT)
}
```

- Struktura reprezentująca jeden z podanych typów, zależnie od spełnionych warunków. Może być parametryzowana wieloma zmiennymi, lub stałymi. Reprezentuje pierwszy typ dla którego spełniony jest warunek. Jeśli żaden warunek nie jest spełniony, przyjmuje typ DEFAULT. Ostatnim typem zawsze musi być DEFAULT.

```
SEQUENCE | SEQUENCE[X1 X2 ...]
{
id TYPE
id2 TYPE2
...
}
```

- Prosta struktura przechowująca zmienne różnych typów. Może być parametryzowana wieloma zmiennymi lub stałymi lub może być bez parametrów. Zmienne mogą być typów wcześniej zadeklarowanych, także parametryzowane

Stałe

```
TRUE, FALSE, LICZBY
```

Zmienne

```
id1, id2...
```

- Zmienne wcześniej zadeklarowane

Deklaracje

Deklaracja struktury danych

```
id ::= TYPE[X1 X2 ...] { ... }
```

- Deklaracja struktury tworzy nową strukturę danego typu (może to być typ prosty), której możemy używać w kolejnych deklaracjach struktur lub bezpośrednio wczytać do niej dane z pliku i przedstawić wynik

Deklaracja zmiennej

```
id TYPE
```

- Deklaracja zmiennej występuje jedynie w środku struktur SEQUENCE i ARRAY, także przekazywane parametry pełnią rolę zmiennych. Nie można zadeklarować ich poza strukturami oraz nie można bezpośrednio do zmiennych wczytywać danych

Uwaga: Nie ma możliwości deklaracji struktur rekurencyjnych, ponieważ podczas deklaracji struktury, ona sama jeszcze siebie nie widzi.

Struktury są deklarowane w podanej kolejności, aby używać struktury musi być ona wcześniej w całości zadeklarowana. Wyklucza to jakąkolwiek możliwość rekurencyjnego deklarowania struktur!

Proste przykłady

Struktura opisująca liczbę 8-bajtową

```
longUint ::= UINT_64
```

Struktura pusta(nie wczytuje danych)

```
empty ::= BITSTRING_0
```

Struktura opisująca dane wymiarów prostokąta

```
Rescangle ::= SEQUENCE
{
	width UINT
	height UINT
}
```

Struktura opisująca kwadrat

```
Square ::= SEQUENCE
{
	side UINT
}
```

Struktura opisująca tablicę prostokątów

```
Rescangles ::= ARRAY[a]
{
	rect Rectangle
}
```

Struktura opisująca kwadrat lub prostokąt, zależnie od parametru

```
RectOrSquare ::= CHOICE[a]
{
	Rectangle(a == TRUE)
	Square(DEFAULT)
}
```

Struktura opisująca tablicę figur(prostokątów lub kwadratów)

```
Figures ::= ARRAY[a]
{
	isRectangle BOOL
	figure RectOrSquare[isRectangle]
}
```