Zyntex

Linguagem de Programação

O que é Zyntex

Propósito

Oferecer aos estudantes de programação uma forma prática e direta para experimentar e consolidar seu entendimento dos conceitos fundamentais dos paradigmas de programação.

Objetivo

- Ajudar os estudantes a comparar diferentes abordagens para a resolução de problemas.
- Fazer com que os estudantes apreciem as vantagens de cada estilo de programação (imperativo e funcional).

Paradigmas

- Paradigma imperativo: variáveis, atribuições e controle de fluxo.
- Paradigma funcional: funções de primeira classe, imutabilidade e funções de ordem superior.

Elementos Principais: Tokens

01 Tipos de Dados

Numero Inteiro	**int**
Numero Decimal	**float**
Caractere	**char**
Cadeia de Caracteres	**string**
Boolean	**bool**
Array	**arr**
Lista	**list**
Função	**function**
Nulo	**null**
Vazio (void)	**void**
Dicionário	**dict**

02 Operadores Unários

Incremento	**+>**
Decremento	**->**

03 Operadores Aritméticos

Soma	**++**
Subtração	**--**
Multiplicação	******
Divisão	**//**
Radiciação	**rd**
Exponenciação	**exp**

04 Operadores Relacionais

Igual a	**=?**
Diferente de	**!=?**
Menor que	**<<**
Maior que	**>>**
Menor ou igual	**<=**
Maior ou igual	**>=**

05 Operadores de Atribuição

Atribuição de Valor	**:=**
Atribuição de Tipo	**=>**

Elementos Principais: Tokens

06 Operadores Lógicos

AND	**&&**
OR	**||**
NOT	**!**

07 Símbolos

Delimitador de Início de Bloco	**{**
Delimitador de Fim de Bloco	**}**
Início de comentário	**##**
Parênteses	**( )**

08 Palavras-chave

Definir função	**fx**
Definir variável	**vx**
Condicional “if”	**zf**
Condicional “else”	**zl**
Laço “for”	**fr**
Laço “while”	**wl**
Impressão	**out**
Entrada de dados	**in**
Retorno de função	**rx**
Valor booleano verdadeiro	**true**
Valor booleano falso	**false**

Exemplo de Código em Zyntex

```
## Exemplo de verificação de idade em Zyntex
vx idade => int
idade := 25

## Estrutura condicional para verificar a idade
zf (idade >> 18) {
  out("Maior de idade")
} zl {
  out("Menor de idade")
}
```

Regras Lexicais e de Formação de Tokens

Identificadores

Nomes de variáveis, funções, etc.

- Devem começar com uma letra (a-z, A-Z) ou underscore (_).
- Podem ser seguidos por qualquer combinação de letras, números (0-9) e underscores.
- São case-sensitive (minhaVar é diferente de minhavar).
- Exemplos: soma, _valor, calculo1

Literais

Representação de valores fixos no código.

- Inteiro (int): Uma sequência de um ou mais dígitos. Ex: 10, 42, 999.
- Decimal (float): Uma sequência de dígitos, um ponto (.), e outra sequência de dígitos. Ex: 3.14, 0.5, 100.0.
- Cadeia de Caracteres (string): Qualquer sequência de caracteres entre aspas duplas ("). Ex: "Olá mundo", "Zyntex".
- Caractere (char): Um único caractere entre aspas simples ('). Ex: 'A', 'z', '7'.
- Booleano (bool): As palavras-chave true e false.
- Nulo (null): A palavra-chave null.

Comentários

- Um comentário é iniciado com ## e continua até o final da linha.
- O analisador léxico deve ignorar completamente os comentários.
- Exemplo: vx idade := 18 ## Define a maioridade penal

Espaçamento (Whitespace)

- Espaços ( ), tabulações (\t) e quebras de linha (\n) são usados para separar tokens.
- Múltiplos espaços são tratados como um só separador e são ignorados pelo analisador (exceto para contar o número das linhas).

Ambiguidade e "Casamento Mais Longo" (Longest Match)

- O analisador deve sempre tentar formar o token mais longo possível a partir da entrada.
- Ao encontrar -, ele deve verificar se o próximo caractere é > (formando ->) ou - (formando --).
- Ao encontrar !, ele deve verificar se os próximos são =? (formando !=?). Se não, o token é apenas !.
- Uma sequência de letras como out deve primeiro ser comparada com a lista de palavras-chave. Se não for nenhuma delas, será classificada como um IDENTIFICADOR.

Conceitos Analisador Léxico

Processo de Análise

Código Fonte
Sequencia de Caracteres do programa Zyntex

Análise Léxica
Reconhecimento de Padrões e Classificação

Tokens
Unidades Léxicas classificadas e estruturadas

Token

Unidade léxica básica com tipo, valor e posição no código fonte.

Reconhecimento de Padrões

Identificação de sequências de caracteres que formam tokens válidos.

Classificação

Categorização dos tokens em tipos específicos da linguagem.

Tratamento de Erros

Detecção e relatório de caracteres ou sequências inválidas.