PurpleKai — Linguagem de Programação Interpretada com Palavras-chave em Francês

Miguel Santana da Costa

Introdução

A linguagem **PurpleKai** é uma linguagem de programação interpretada, orientada a objetos, com tipagem dinâmica forte. Inspirada na sintaxe do C++ e Python, ela utiliza **palavras-chave em francês**, tornando a escrita mais natural para falantes do idioma. A linguagem foi projetada com os seguintes princípios:

- **Acessibilidade**: Sintaxe clara e legível para falantes de francês
- **Interoperabilidade**: Compatibilidade com bibliotecas C++ através de FFI
- **Produtividade**: Tipagem dinâmica com inferência de tipos
- **Segurança**: Gerenciamento automático de memória com coletor de lixo

Mapeamento de Palavras-Chave: PurpleKai ↔ C++

Palavras-chave da Linguagem

Palavra-chave (FR)	Descrição (PT)	Description (EN)
classe	Declara uma classe orientada a objetos	Declares an object-oriented class
structure	Define uma estrutura simples	Defines a simple struct
public	Membro acessível publicamente	Publicly accessible member
prive	Membro privado, acessível apenas internamente	Private member, accessible only internally
protégé	Membro protegido, visível por subclasses	Protected member, visible by subclasses
si	Declaração condicional (se)	Conditional statement (if)
sinon	Bloco alternativo à condição (senão)	Alternative block (else)
tantque	Laço de repetição enquanto a condição for verdadeira	Loop while condition is true

Palavra-chave (FR)	Descrição (PT)	Description (EN)
pour	Laço com contador e condição de parada	For loop with counter and stop condition
interrompre	Interrompe um laço de repetição	Breaks a loop iteration
continuer	Avança para a próxima iteração do laço	Skips to the next loop iteration
retourner	Retorna um valor de uma função	Returns a value from a function
fonction	Declara uma função	Declares a function
nouveau	Instancia um novo objeto dinamicamente	Instantiates a new object dynamically
ceci	Referência ao próprio objeto (this)	Reference to the current object (this)
nul	Representa valor nulo ou ausência de objeto	Represents null value or no object
vrai	Valor booleano verdadeiro	Boolean value true
faux	Valor booleano falso	Boolean value false
laisser	Declara uma variável mutável	Declares a mutable variable
constant	Declara uma constante (imutável)	Declares an immutable constant
afficher	Envia saída para o console	Outputs data to console
lire	Lê entrada do usuário	Reads input from user
importer	Importa um módulo ou biblioteca	Imports a module or library
essayer	Inicia um bloco de tratamento de erro	Starts a try block for error handling
attraper	Captura uma exceção lançada	Catches a thrown exception
lancer	Lança uma nova exceção	Throws an exception
dans	Utilizado em laços do tipo ”para item em lista”	Used in loops like ”for item in list”
objet	Referência genérica a um objeto instanciado	General reference to an instantiated object

Sistema de Tipos

PurpleKai implementa um sistema de tipos dinâmico com as seguintes características:

$$\text{Tipos} = \{\text{Entier}, \text{Réel}, \text{Booléen}, \text{Chaîne}, \text{Liste}, \text{Dictionnaire}, \text{Objet}, \text{Fonction}, \text{Nul}\}$$
$$\text{Operações} = \{+, -, *, /, \%, ==, !=, <, >, \&\&, ||\}$$
$$\text{Conversões implícitas somente quando seguras}$$

Tabela 1: Correspondência entre palavras-chave da PurpleKai (francês) e C++ (inglês)

PurpleKai	**C++**	**Tipo**	**Descrição Detalhada**
`classe`	`class`	Declaração	Define uma classe. Em PurpleKai, todas as classes são implicitamente herdadas de `Objet`
`hérite`	`: public`	Herança	Estabelece herança simples. Suporta herança múltipla: `classe C hérite A, B`
`fonction`	-	Função	Opcional para métodos. Pode ser omitida: `maMéthode() {...}`
`constructeur`	`[ClassName]`	Método	Pode ter vários construtores. Chamado com `nouveau`
`self`	`this`	Ponteiro	Referência implícita em métodos. Ex: `self.x = 10`
`super`	`::`	Acesso	Acesso a membros da superclasse: `super.méthode()`
`laisser`	`auto`	Variável	Inferência de tipo dinâmico: `laisser x = 10`
`constant`	`const`	Constante	Imutável após declaração: `constant PI = 3.14`
`si`	`if`	Controle	Suporta `sinon si` para `else if`
`pour`	`for`	Loop	Versão range-based: `pour i dans 1..10`
`interface`	`abstract`	Interface	Métodos puramente virtuais: `fonction virtuelle()`

Exemplo de Tipagem Dinâmica

```
laisser x = 10          # Entier
x = "Bonjour"           # Cha ne
x = [1, 2, 3]           # Liste

fonction ajouter(a, b) {
    retourner a + b     # Polymorphisme selon les types
}

ajouter(5, 10)          # 15
ajouter("a", "b")       # "ab"
```

Listing 1: Exemplo PurpleKai

Programação Orientada a Objetos

Exemplo Comparativo

```
classe Animal {
  fonction constructeur(n) {
    self.nom = n;
  }

  fonction parler() {
    retourner "??";
  }
}

classe Chien h rite Animal {
  fonction parler() {
    retourner "Woof!";
  }
}

laisser chien = nouveau Chien("Rex
   ")
afficher(chien.parler())
```

Listing 2: PurpleKai

```
class Animal {
public:
  Animal(string n) : nom(n) {}

  virtual string parler() {
    return "??";
  }
};

class Chien : public Animal {
public:
  string parler() override {
    return "Woof!";
  }
};

Chien* chien = new Chien("Rex");
cout << chien->parler();
delete chien;
```

Listing 3: C++

Polimorfismo Avançado

```
interface Parlant {
    fonction parler()
}

classe Humain impl mente Parlant {
    fonction parler() {
        retourner "Bonjour!"
    }
}

classe Perroquet impl mente Parlant {
    fonction parler() {
        retourner "Coco veut un cracker!"
    }
}

fonction faireParler(Parlant p) {
    afficher(p.parler())
}

faireParler(nouveau Humain())   # Bonjour!
faireParler(nouveau Perroquet()) # Coco veut un cracker!
```

Listing 4: Exemplo de Polimorfismo

Features Avançadas

Programação Funcional

```
# Fun  es de alta ordem
laisser nombres = [1, 2, 3, 4, 5]

# Filtro com fun  o an nima
laisser pairs = nombres.filtrer(fonction(x) {
    retourner x % 2 == 0
})

# Redu  o
laisser somme = nombres.r duire(0, fonction(acc, x) {
    retourner acc + x
})

afficher(pairs)  # [2, 4]
afficher(somme)  # 15
```

Tratamento de Erros

```
fonction lireFichier(chemin) {
    essayer {
        laisser fichier = ouvrir(chemin)
        laisser contenu = fichier.lire()
        retourner contenu
    } attraper (ErreurIO e) {
        afficher("Erreur de lecture:", e.message)
        retourner ""
    } enfin {
        afficher("Traitement termin ")
    }
}
```

Observações Importantes

- **Tipagem**: PurpleKai usa tipagem dinâmica forte vs C++ com tipagem estática
- **Memória**: Gerenciamento automático vs manual em C++
- **Sintaxe**: Mais concisa que C++ com menos símbolos especiais
- **Performance**: Interpretada (mais lenta) vs compilada (C++)
- **Paradigmas**: Suporte nativo a OO e funcional vs principalmente OO em C++

Conclusão

PurpleKai oferece uma alternativa moderna que combina:

- A expressividade de linguagens dinâmicas como Python
- A estrutura de linguagens OO como C++/Java
- A acessibilidade para falantes de francês
- Segurança com gerenciamento automático de recursos

Ideal para:

- Educação em programação
- Prototipagem rápida
- Scripting de alto nível
- Desenvolvimento ágil

Documentação completa e exemplos disponíveis em:
https://github.com/MiguelSantanaDaCosta/PurpleKai/blob/main/PurpleKai.pdf