SYSTÈME EXPERT: LO21

CONSTANT JULIEN & ECHARD NOE

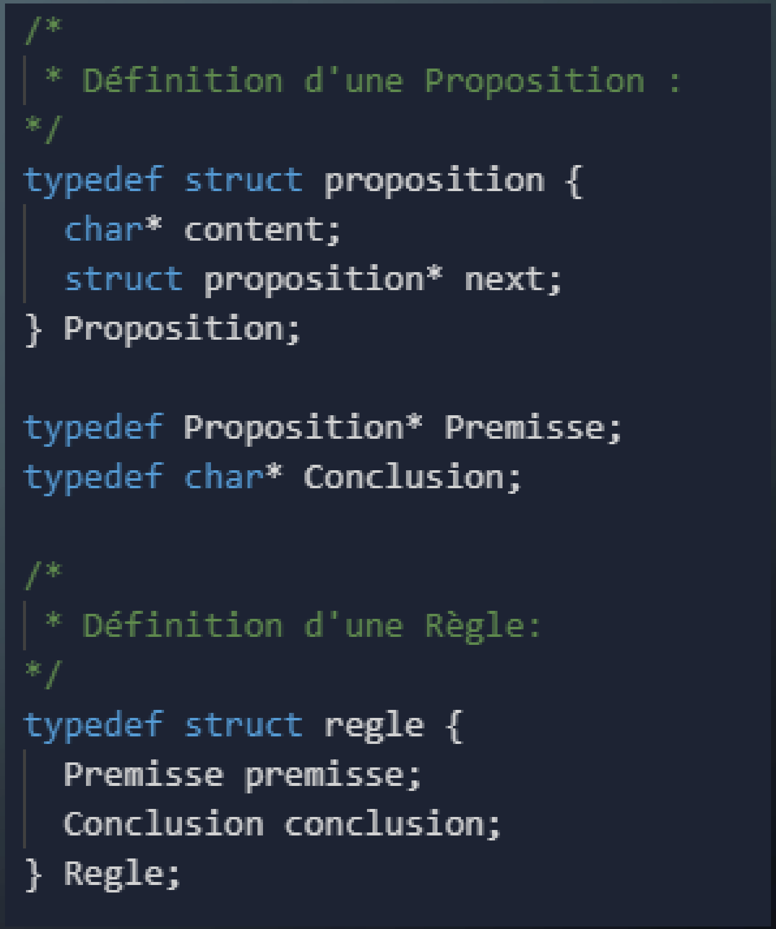
STRUCTURES UTILISÉES:

- Règles:

Contient une liste chainée de prémisse et une conclusion.

- Prémisse / Proposition:

Liste chainée contenant des strings.



```c
/*
 * Définition d'une Proposition :
*/
typedef struct proposition {
  char* content;
  struct proposition* next;
} Proposition;

typedef Proposition* Premisse;
typedef char* Conclusion;

/*
 * Définition d'une Règle:
*/
typedef struct regle {
  Premisse premisse;
  Conclusion conclusion;
} Regle;
```

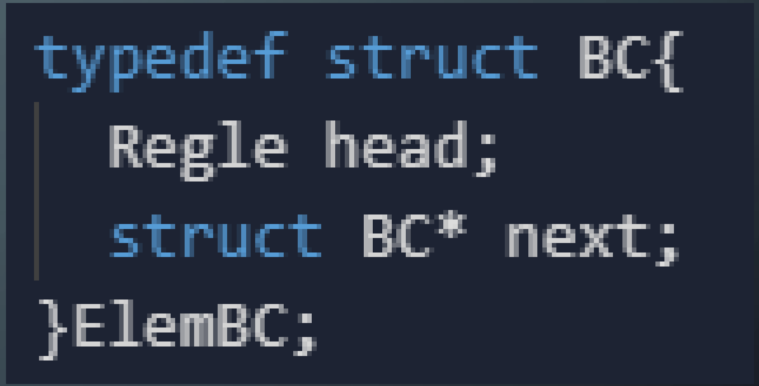
STRUCTURES UTILISÉES:

- Base de Connaissance:

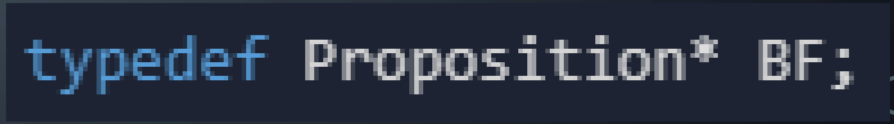
Une liste chainée règles.

- Base de Faits:

Une liste chainée contenant des strings.

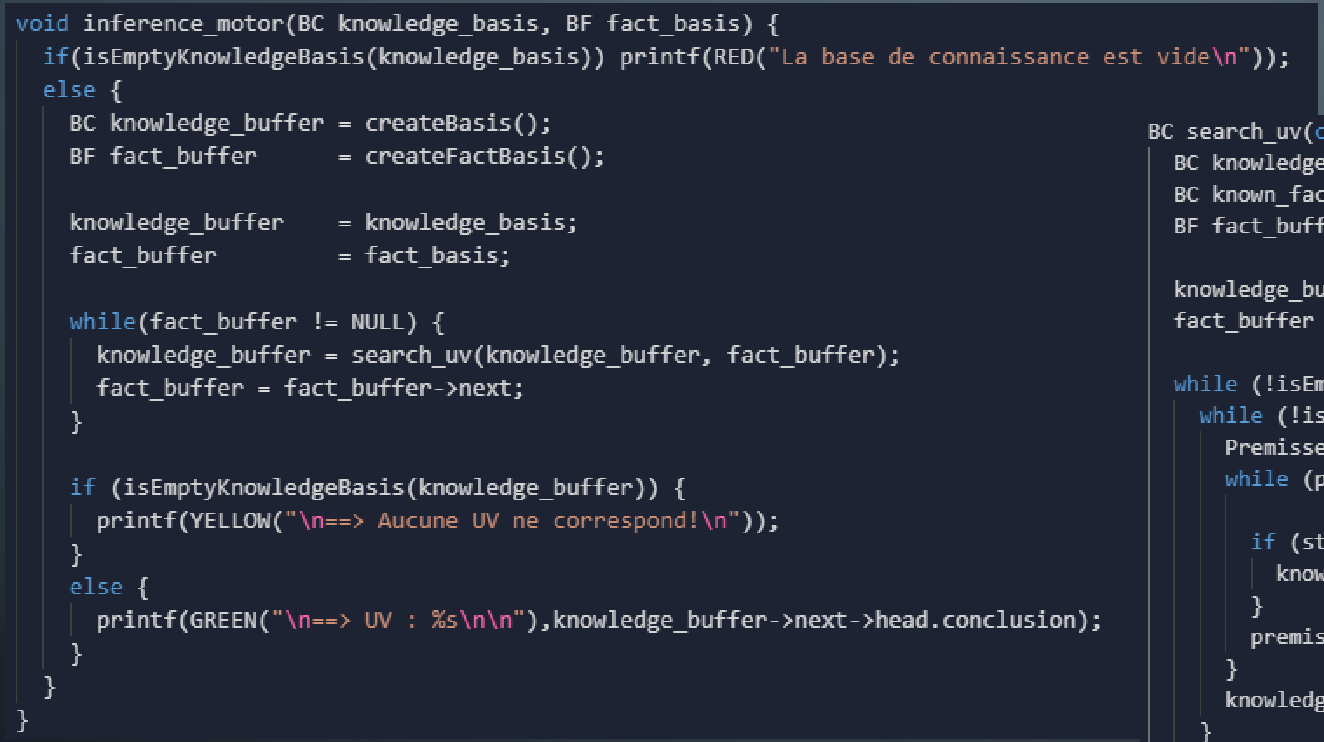


```
typedef struct BC{
    Regle head;
    struct BC* next;
}ElemBC;
```



```
typedef Proposition* BF;
```

MOTEUR D'INFÉRENCE:

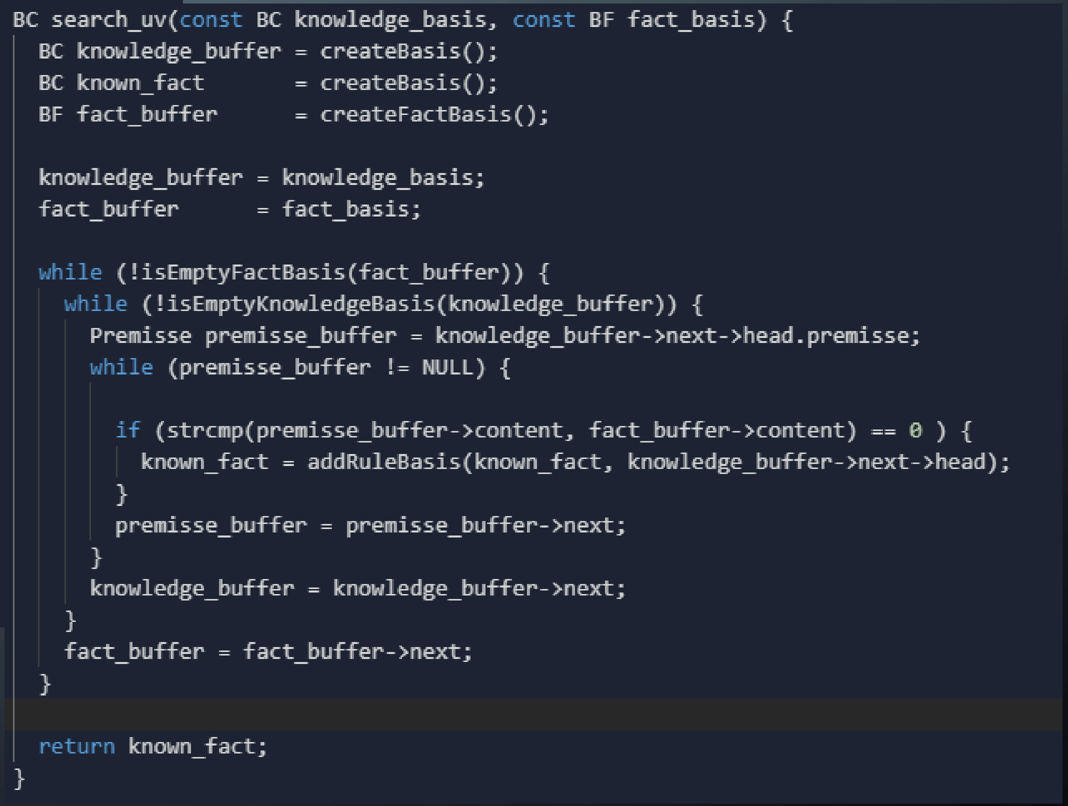


```c
void inference_motor(BC knowledge_basis, BF fact_basis) {
  if(isEmptyKnowledgeBasis(knowledge_basis)) printf(RED("La base de connaissance est vide\n"));
  else {
    BC knowledge_buffer = createBasis();
    BF fact_buffer      = createFactBasis();

    knowledge_buffer    = knowledge_basis;
    fact_buffer         = fact_basis;

    while(fact_buffer != NULL) {
      knowledge_buffer = search_uv(knowledge_buffer, fact_buffer);
      fact_buffer = fact_buffer->next;
    }

    if (isEmptyKnowledgeBasis(knowledge_buffer)) {
      printf(YELLOW("\n==> Aucune UV ne correspond!\n"));
    }
    else {
      printf(GREEN("\n==> UV : %s\n\n"),knowledge_buffer->next->head.conclusion);
    }
  }
}
```



```c
BC search_uv(const BC knowledge_basis, const BF fact_basis) {
  BC knowledge_buffer = createBasis();
  BC known_fact       = createBasis();
  BF fact_buffer      = createFactBasis();

  knowledge_buffer = knowledge_basis;
  fact_buffer      = fact_basis;

  while (!isEmptyFactBasis(fact_buffer)) {
    while (!isEmptyKnowledgeBasis(knowledge_buffer)) {
      Premisse premisse_buffer = knowledge_buffer->next->head.premisse;
      while (premisse_buffer != NULL) {

        if (strcmp(premisse_buffer->content, fact_buffer->content) == 0 ) {
          known_fact = addRuleBasis(known_fact, knowledge_buffer->next->head);
        }
        premisse_buffer = premisse_buffer->next;
      }
      knowledge_buffer = knowledge_buffer->next;
    }
    fact_buffer = fact_buffer->next;
  }

  return known_fact;
}
```

CONCLUSION:

Merci de votre attention.