RAPPORT SYSTEME EXPERT

Realisé par :

IDM'BARK LOUBNA

KERBOUB hAMZA

DEFINITION :

Un **système expert** est un outil capable de reproduire les mécanismes cognitifs d'un expert, dans un domaine particulier. Il s'agit de l'une des voies tentant d'aboutir à l'intelligence artificielle.

Plus précisément, un système expert est un logiciel capable de répondre à des questions, en effectuant un raisonnement à partir de faits et de règles connues. Il peut servir notamment comme outil d'aide à la décision. Le premier système expert a été Dendral. Il permettait d'identifier les constituants chimiques.

Composition :

Un système expert se compose de 3 parties :

- une base de faits ;
- une base de règles ;
- un moteur d'inférence.

Le moteur d'inférence est capable d'utiliser faits et règles pour produire de nouveaux faits, jusqu'à parvenir à la réponse à la question experte posée.

La plupart des systèmes experts existants reposent sur des mécanismes de logique

formelle et utilisent le raisonnement déductif. Pour l'essentiel, ils utilisent des **règles d'inférence** de la forme suivante (syllogisme) :

si P est vrai (fait ou prémisse) et si on sait que P implique Q (règle) alors, Q est vrai (nouveau **fait** ou **conclusion**).

Les plus simples des systèmes experts s'appuient sur la logique des propositions (dite aussi « logique d'ordre 0 »). Dans cette logique, on n'utilise que des propositions, qui sont vraies, ou fausses. D'autres systèmes s'appuient sur la logique des prédicats du premier ordre (dite aussi « logique d'ordre 1 »), que des algorithmes permettent de manipuler aisément.

Il faut maintenir une certaine cohérence de l'ensemble des règles:

Incompatibilité (R1: Si A et B alors C; R2: Si A et B alors D;)

Redondance (R1: Si A alors B; R2 : Si C alors B; Sauf si on applique des coefficients de certitude différents)

Bouclage (R1: Si A alors B; R2: Si B alors C; R3:

Si C alors A;)

Enfin, pour faciliter la description de problèmes réels sous forme de règles logiques, on a recours à des opérateurs ou des valeurs supplémentaires (notions de nécessité/possibilité, coefficients de plausibilité, etc.)

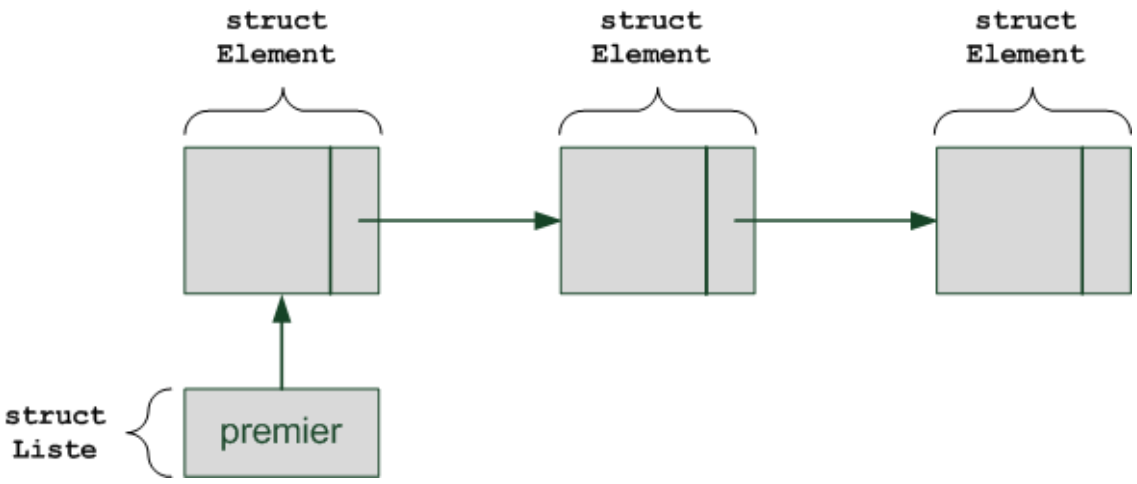
Complexité des systèmes :

En pratique, dès que l'on dépasse la centaine de règles, il devient très difficile de suivre comment le système expert « raisonne » (manipule faits et règles en temps réel), et donc d'en assurer la mise au point finale, puis la maintenance

NOTRE PROJET :

L’objectif de ce projet est d'implementer en **C (ou C + +)** un outit qui permet de créer **des systemes experts** modestes et de l’utiliser pour créer un systeme expert pour recommender un langage de programmation pour un apprenant. L 'outil à realiser doit etre implementer en utilisant **les listes chainées .**

- **Listes chainées :**



Une liste chaînée ou liste liée (en anglais linked list) désigne en informatique une structure de données représentant une collection ordonnée et de taille arbitraire d'éléments de même type, dont la représentation en mémoire de l'ordinateur est une succession de cellules faites d'un contenu et d'un pointeur vers une autre cellule. De façon imagée, l'ensemble des cellules ressemble à une chaîne dont les maillons seraient les cellules.

L'accès aux éléments d'une liste se fait de manière séquentielle : chaque élément permet l'accès au suivant (contrairement au tableau dans lequel l'accès se fait de manière directe, par adressage de chaque cellule dudit tableau).

IMPLEMENTATION EN C

·4 LES LISTES

On a crée une structure "**Reponse**" qui contient

comme donnée la reponse et qui pointe sur la question suivante .

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

typedef struct Reponse{
    char rps[50];
    struct Question *suivant;
}Reponse;
```

Puis on a déclarer une deuxieme liste "**Question**" qui a comme donnée une variable de type char contient la question et un tableau de type **Reponse** qui contient l'ensemble des reponses possible .

```c
typedef struct Question{
    char qst[50];
    struct Reponse tab[10];
    int p;
}Question;
```

Apres on a crée une liste "**regle**" qui permet la gestion des regles contenant la reponse et la regle sur laquelle elle pointe .

```c
typedef struct Regle{
    struct Reponse *p;
    struct Regle *next;
}Regle;
```

Liste "**Base_regle** " qui traite les regles afin de donner une conclusion .

```
typedef struct Base_Regle{
    struct Regle *r;
    struct Base_Regle *next;
    char conclusion[10];
}Base_Regle;
```

Puis liste "**User_Answer**"qui stock les repponses entrées par l'utilisateur .

```
typedef struct User_Answer{
    char answer[50];
    struct User_Answer *next;
}User_Answer;
```

·5 LES FONCTIONS

Dans notre on a crée plusieurs fonctions afin d'assurer la bonne resolution de notre probleme :

Fonction pour créer une question

```
Question* Creer_Question(){
    Question *b;
    b=NULL;
    return b;
```

Fonction pour ajouter une question

```
Question* Ajouter_Question(char question[]){
    Question *d=Creer_Question();
    d=(Question *)malloc(sizeof(Question));
    strcpy(d->qst,question);
    d->p=0;
    return d;
}
```

Fonction pour créer une reponse

```
Reponse* Creer_Reponse(){
     Reponse *b;
     b=NULL;
     return b;
}
```

Fonction pour ajouter une reponse

```
Reponse* Ajouter_Reponse(Question *question,char reponse[]){
    Reponse *b=Creer_Reponse();
    b=(Reponse *)malloc(sizeof(Reponse));
    b->suivant=question;
    strcpy(b->rps,reponse);
    return b;
}
```

Fonction pour affecter une reponse à une question

```
void Ajouter_reponse_question(Question *head1,Reponse *head2,int pos){
    head1->tab[pos]=*head2;
    head1->p++;
}
```

Une fonction qui crée une regle et une autre qui lui affecte une reponse :

```
Regle* Creer_Regle(){
    Regle *b;
    b=NULL;
    return b;
}

Regle * Ajouter_Regle(Regle *r,Reponse *t1){
    Regle *b;
    b=(Regle *)malloc(sizeof(Regle));
    b->p=t1;
    b->next=NULL;
    if(r==NULL)
        r=b;
    else{
        Regle *temp=r;
        while(temp->next!=NULL)
            temp=temp->next;
        temp->next=b;
    }
    return r;
}
```

Une fonction qui crée une Base et une autrre fonction qui lui affecte des regles :

```
Base_Regle* Creer_Base_Regle(){
    Base_Regle *b;
    b=NULL;
    return b;
}

Base_Regle * Ajouter_Base_Regle(Base_Regle *br,Regle *r1,char conc[]){
    Base_Regle *b;
    b=(Base_Regle *)malloc(sizeof(Base_Regle));
    b->r=r1;
    strcpy(b->conclusion,conc);
    b->next=NULL;
    if(br==NULL)
        br=b;
    else{
        Base_Regle *temp=br;
        while(temp->next!=NULL)
            temp=temp->next;
        temp->next=b;
    }
    return br;
}
```

La fonction Creer_User_Answer et ajouter_User_Answer :

```c
User_Answer* Creer_User_Answer(){
    User_Answer *b;
    b=NULL;
    return b;
}

User_Answer * Ajouter_User_Answer(User_Answer *r,char p[]){
    User_Answer *b;
    b=(User_Answer *)malloc(sizeof(User_Answer));
    strcpy(b->answer,p);
    b->next=NULL;
    if(r==NULL)
        r=b;
    else{
        User_Answer *temp=r;
        while(temp->next!=NULL)
            temp=temp->next;
        temp->next=b;
    }
    return r;
}
```

Fonction pour creer L'interface :

```c
User_Answer * Interface(Question *q){
    int i,flag=0;
    int answer;
    User_Answer *Answers=Creer_User_Answer();
    Question *temp;
    printf("%s \n",q->qst);
    for(i=0;i<q->p;i++){
        printf("%d) %s \n",i+1,q->tab[i].rps);
    }
    do{
    flag=0;
    printf("Entrer une des reponses :");
    scanf("%d",&answer);
    for(i=0;i<q->p;i++){
        if(answer-1 == i) {flag=1;}
    }
    }while(flag!=1);
    Answers=Ajouter_User_Answer(Answers,q->tab[answer-1].rps);
    temp=q->tab[answer-1].suivant;
```

```
    while((temp->qst)!=NULL){
        printf("\n%s \n",temp->qst);
        for(i=0;i<temp->p;i++){
            printf("%d) %s \n",i+1,temp->tab[i].rps);
        }
    do{
        flag=0;
        printf("Entrer une des reponses :");
        scanf("%d",&answer);
        for(i=0;i<temp->p;i++){
            if(answer-1 == i) {flag=1;}
        }
        }while(flag!=1);
        Answers=Ajouter_User_Answer(Answers,temp->tab[answer-1].rps);
        temp=temp->tab[answer-1].suivant;

    }
    return Answers;
}
```

Moteur d'inference :

```
void Moteur_Inference(User_Answer *R,Base_Regle *B){
    User_Answer *temp=R;
    Base_Regle *temp1=B;
    Regle *temp2=Creer_Regle();
    int x;
    while (temp1!= NULL){
        temp2=temp1->r ;
        x=0;
        while (temp!=NULL || temp2!= NULL ){
            if (strcoll(temp->answer , temp2->p->rps)==0){
                temp=temp->next ;
                temp2=temp2->next ;
            }
            else { x=1;  break ;}
        }
        if (x==0){
            printf(temp1->conclusion );
        }

        else {
            temp1=temp1->next ;
            temp=R ;

        }
}

}
```

La fonction "main()" : Dans la fonction main() de notre programme on a declaré une suite d'instructions qui permettent la realisation de notre objectif , on a commenceé par l'insertion des questoins à poser , l'insertion des reponses possibles de chaque

question , la creation des regles et les bases de regles et finallement l'appel du moteur d'inference .

voici quelques exemples :

Insertion questions :

```
int main(){
    Question *question1=Ajouter_Question("why do you want to learn programming?");
    Question *question2=Ajouter_Question("Do you have any idea ?");
    Question *question3=Ajouter_Question("how do you prefer to learn things ?");
    Question *question4=Ajouter_Question("auto or manuel?");
    Question *question5=Ajouter_Question("wich platforme ?");
    Question *question6=Ajouter_Question("does your web app provid infi in real time like twitter ?");
    Question *question7=Ajouter_Question("do you want to try something new ?");
    Question *question8=Ajouter_Question("which one is your favorite ?");
    Question *question9=Ajouter_Question("which OS ?");
```

Insertion Reponses :

```
    Reponse *reponse1=Ajouter_Reponse(NULL,"pick");
    Reponse *reponse2=Ajouter_Reponse(NULL,"kids");
    Reponse *reponse3=Ajouter_Reponse(question2,"fun");
    Reponse *reponse5=Ajouter_Reponse(question13,"improve My Self");
    Reponse *reponse6=Ajouter_Reponse(question11,"make Money");
```

Affecter reponse a questions :

```
    Ajouter_reponse_question(question1,reponse1,0);
    Ajouter_reponse_question(question1,reponse2,1);
    Ajouter_reponse_question(question1,reponse3,2);
    Ajouter_reponse_question(question1,reponse5,3);
    Ajouter_reponse_question(question1,reponse6,4);
```

Creer regle et base :

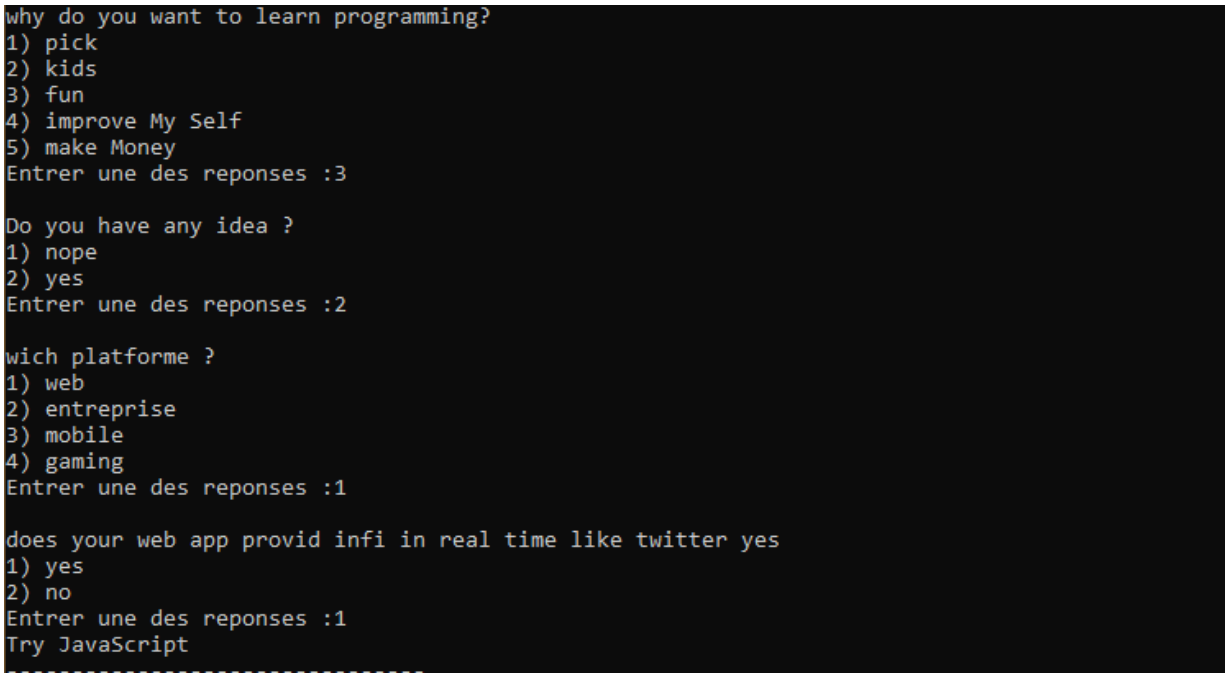
```
Regle *regle36 = Creer_Regle ();


regle1 = Ajouter_Regle (regle1, reponse1);
Base_Regle *Base1 = Creer_Base_Regle ();
Base1 = Ajouter_Base_Regle (Base1, regle1, "Python");
```

Moteur d'inference :

```
  User_Answer *User_reponses = Creer_User_Answer ();
  User_reponses = Interface (question1);
  Moteur_Inference (User_reponses, Base1);
}
```

Exemple d'execution :



Conclusion :

Lors de la realisation de ce travail on a eu l'occasion de découvrir un logiciel tres puissant capable de reproduire les mécanismes cognitifs d'un expert, dans un domaine particulier.... ce qui va nous aider à enrichir nos connaissances et savoir faire dans notre domaine .