Trabalho realizado pelo grupo lfa-08:

93284 – Miguel Gomes Fernandes

93295 – João Diogo Oliveira

93301 – Ana Luísa Ferreira

93305 – João Diogo Ferreira

Tema: questionário.

Organização

O trabalho está localizado na pasta Main. Nela encontram-se as duas gramáticas produzidas, o compilador, o analisador semântico, o ficheiro Main produzido e o visitor da linguagem da base de dados. Também contém os seguintes diretórios:

- Demo: contém ficheiros de programas funcionais.
- javaClasses: contém as classes criadas de apoio ao programa.
- javaTypes: contém as classes dos tipos.
- qstFiles: contém os ficheiros das bases de dados.
- Templates: contém os ficheiros ".stg" usados.
- TestsErros: contém exemplos de erros de programas.

Como correr o programa:

Na pasta Main:

- Fazer antlr4-build duas vezes. (i.e. a primeira cria os ficheiros relativos a gramática da base de dados e a segunda vez consegue compilar a gramática principal)
- java QuestionarioMain.java <ficheiro> (exemplo: java QuestionarioMain.java TestsErros/error0.txt **OU** java QuestionarioMain.java Demo/simples.txt)
- javac Output.java
- java Output

Linguagem da base de dados

A linguagem mais simples implementada, serve para definir uma base de dados.

Para criar um ficheiro para a base de dados é preciso definir o ficheiro com a extensão ".qst".

Primeiro são definidos os limites da dificuldade da base de dados e é atribuída uma dificuldade default, usada quando não for redefinida.

Depois são declaradas as variáveis com a cotação associada, se esta não existir dá erro.

A definição da base de dados está organizada por temas, podem existir temas dentro de temas e por fim perguntas dentro de temas. Tanto no tema como nas perguntas a dificuldade pode ser redefinida.

As perguntas têm de ser definidas com um identificador único Px (exemplo: P1). E dentro de cada pergunta têm de existir pelo menos uma resposta identificada com uma alínea, sendo estas todas diferentes também, por exemplo:

a – “conteúdo da alínea”: <variável>

A cotação de cada resposta é definida com a atribuição das variáveis criadas ao início.

```
difficulty[0:10] := 5;

V = 100;
F = 0;

<"Cultura geral">
        <"Historia de portugal" := 4 >

                P1("Em que século Portugal se tornou um país independente?") {
                        a - "XII":V;
                        b - "XIV":F;
                        c - "XIII":V;
                        d - "Nenhum dos anteriores":F;
                }

                P2("Ministro de D. José I, Sebastião José de Carvalho e Melo ficou também conhecido como? ") {
                        a - "Marquês de Pombal.":V;
                        b - "Duque da Terceira.":F;
                        c - "Marquês de Fronteira.":F;
                        d - "Nenhuma das anteriores":F;
                }


        </"Historia de portugal">


        <"Gramática">

                <"Classe de palavras" := 7>
                        P7("Qual das frases tem dois advérbios?") {
                                a - "Talvez fossem boas pessoas.":F;
                                b - "Ali ninguém conhecia bem os seus segredos.":V;
                                c - "Trabalhavam muito durante a tarde.":F;
                                d - "Não receavam a maldade daquelas pessoas.":F;
                        }

                </"Classe de palavras">

        </"Gramática">

</"Cultura geral">
```

Exemplo de como construir uma base de dados

Linguagem do compilador

Para ter acesso ás perguntas de um base de dados basta usar a instrução load:

load <ficheiroBaseDadosPath> as <nome>

nota:

- o path deve ser definido em relação ao ficheiro qstFiles (por uma questão de conveniência)
- os ficheiros devem ter a extensão .qst
- no programa os ficheiros são referenciados por <nome>

Quiz e o bloco onde é definida a estrutura do Quiz.

Bloco Quiz:

- Estrutura:

```
Quiz <Name>(<Participante>, <score>){
    ...
}
```

Onde <Participante> é uma String e <score> um inteiro (cotação máxima do quiz), a serem atribuídos durante a sua instanciação.

Pode ter um bloco de opções associado:

- Bloco Options:

Conjunto possível de opções, que pode ser adicionado por uma resposta do utilizador.

- Estrutura:

```
Options <OptionsName> (
    <OptionID> : <"OptionText">;
    ...
    <OptionID> : <"OptionText">;
);
```

Durante uma resposta se um o utilizador digitar <"OptionText"> uma ação pode ser desencadeada.

- Bloco Section:

Conjunto de perguntas.

Contém a percentagem a atribuir a cada tema.

- Estrutura:

```
Section <Name><Percentage> (
        <P1>
        ...
);
```

Percentage: percentagem (int%) , 0 <= int <= 100;

No final o total das percentagens deve ser 100%, isto é, a cotação deve estar dividida de forma a que a somada das percentagens de cada secção no final de 100%.

A percentagem é um atributo opcional.

As perguntas podem ser acedidas no programa principal da seguinte forma:

- Perguntas:

<QuizName>.<SectionName>.current : referente à pergunta atual, de uma determinada secção.

<QuizName>.<SectionName>.prev : referente à pergunta anterior, de uma determinada secção.

<QuizName>.<SectionName>.next : referente à pergunta seguinte, de uma determinada secção.

<QuizName>.<SectionName>.start : referente à primeira pergunta, de uma determinada secção.

<QuizName>.<SectionName>.end : referente à última pergunta, de uma determinada secção.

<QuizName>.<SectionName>.<questionID> : referente à pergunta com ID <questionID>, de uma determinada secção.

SectionName pode ser referido como current

Exemplo

<QuizName>.current.current

##

No bloco Main podem ser executadas ações sob as perguntas de determinada Section de uma determinada instância de um quiz (podem devolver resultados).

- Ações (sob Question):

show <Question> : mostra o texto da pergunta associada a determinada pergunta.

theme <Question> [return String] : devolve a String contendo o tema da pergunta.

value <Question> [return double] : devolve o resultado associado a determinada pergunta, relativo à cotação definida.

result <Question> [return double] : devolve o resultado associado a determinada pergunta.

selected <Question> [return String]: devolve o ID da resposta dada, uma string vazia caso não tenho sido atribuída resposta.

Nota: <Question> é equivalente a qualquer referência enumerada anteriormente em Perguntas (ex: <QuizName>.<SectionName>.current).

Podem ser executadas ações sob as Sections de uma determinada instância de um Quiz, permite iterar sobre cada Section.

- Ações (sob Section):

next <QuizName>.<SectionName> : avançar para a próxima pergunta numa determinada Section (<QuizName>.<SectionName>.current passa a referir a pergunta <QuizName>.<SectionName>.next).

nextUanswered <QuizName>.<SectionName> : avançar para a próxima pergunta por responder, numa determinada Section.

prevUnanswered <QuizName>.<SectionName> : recuar para a pergunta anterior por responder, numa determinada Section.

prev <QuizName>.<SectionName> : recuar para a pergunta anterior numa determinada Section.

start <QuizName>.<SectionName> : pergunta atual de uma determinada Section passa a ser a primeira pergunta.

end <QuizName>.<SectionName> : pergunta atual de uma determinada Section passa a ser a última.

goto <QuizName>.<SecttionName><questionID> : pergunta atual de uma determinada Section passa a ser a pergunta com id questionID.

result <QuizName>.<SectionName> [return double] : devolve o resultado obtido na secção.

value <QuizName>.<SectionName> [return double] : devolve o resultado obtido na secção, relativo à cotação definida.

sizeof <QuizName>.<SectionName> [return int] : devolve o tamanho da section.

- Ações (sob Quiz) :

next <QuizName> : avançar para a proxima Section num determinado Quiz (<QuizName>.current passa a referir a Section <QuizName>.next).

prev <QuizName> : recuar para a Section anterior num determinado Quiz.

start <QuizName> : Section atual de um determinado Quiz passa a ser a primeira pergunta.

end <QuizName> : Section atual de um determinado Quiz passa a ser a última.

goto <QuizName>.<SectionName> : Section atual de um determinado Quiz passa a ser a Section com o nome SectionName.

result <QuizName> [return int] : devolve o resultado obtido no Quiz.

sizeof <QuizName>.<SectionName> [return int] : devolve o tamanho do quiz.

Nota: Existem ações comuns a diferentes estruturas (ex: result, pode ser aplicado a Question, a Section e a Quiz)

Bloco Main:

Bloco principal onde é definido o funcionamento das estruturas Quiz instanciadas.

Estrutura:

```
main {
  <Stat>;
  ...
  <Stat>;
}
```

Conjunto de instruções associadas:

- Instanciação de um novo Quiz:

```
Quiz <QuizName> = new <Name>();
```

- Ações sob as Sections e perguntas das mesmas, de uma instância de um quiz, como referido anteriormente.

- Suporte a tipos int, double, boolean e à álgebra associada a cada um destes tipos.

- Suporte a Strings e operações de concatenação '+'.

- Suporte a blocos condicionas (if else) e a blocos iterativos (while).

- Suporte aos comandos de leitura de inputs (read) e escrita na consola (print)

- Instrução answer:

Estrutura:

```
answer(<Question>) {
  case(<Option>):
    ...
    break;
  valid:
    ...
    break;
  invalid:
    ...
    break;
}
```

A instrução answer fica à espera da resposta do utilizador, de seguida em função desta executa um conjunto de instruções (similar ao switch).

A resposta do utilizador pode ser qualquer opção ativa definida no bloco Options do Quiz ou uma determinada alínea valida. No caso de uma resposta inválida o programa executa as instruções definidas em invalid, caso seja uma alínea valida o programa executa as instruções definidas em valid.

-Instrução adicional 'clear' que permite limpar a consola.

Palavras reservadas:

load
Quiz
Options
Section
from
random
all
as
main
Timer
int
double
boolean
String
sizeof
result
current
next
prev
start
end
goto
nextUnaswered
prevUnaswered
show
print
case
valid
invalid
read
enable
if
else if
else
while
for
time
endTime

O que não foi implementado:

No início tivemos várias ideias e não foram todas implementadas, como gerar um questionário com perguntas aleatórias, ou gerar um questionário com base nos temas e também a ideia de utilizar timers para gerir o tempo de um questionário. Os timers não foram implementados porque não dominamos programação concorrente o resto não foi implementado porque não tivemos tempo.

Contribuição dos autores:

Todos os autores do trabalho contribuíram de igual forma o que atribui a cada um 25%.