**Requisitos:**

**Especificação da linguagem Fluxo-Gráfica e ferramentas do IDE**

Pretende-se a criação de uma ferramenta que permita o desenvolvimento e a aprendizagem de algoritmos através de fluxogramas. Um fluxograma é um conjunto de símbolos gráficos, que contêm texto e são unidos por linhas de fluxo. Para este tópico os requisitos serão identificados por “FLUX.xx.xx”.

**FLUX.00** – Requisitos globais da linguagem.

**FLUX.01** - Inicio/Fim.

**FLUX.02** - União.

**FLUX.03** - Leitura.

**FLUX.04** – Processo/Funções.

**FLUX.05** – Decisão.

**FLUX.06** – Escrita.

**FLUX.07** – Return.

**FLUX.08** – Ligações.

**FLUX.09** – Switch Case.

**FLUX.10** – For.

**FLUX.11** - Comentários.

**FLUX.12** - Editor de expressões.

**FLUX.13** - Interacção com página de testes.

**FLUX.14** - Funções.

**Especificação de Parsing**

Pretende-se a especificação de erros devolvidos pelo parser e a forma como o mesmo se comporta com a definição de strings. A especificação está organizada da seguinte forma:

**PARS.00** – Requisitos Globais

**PARS.01** – Linguagens de baixo nível (Assembly)

**PARS.02** – Linguagens de médio/alto nível

**PARS.03** – Fluxograma

**PARS.04** –Strings

**PARS.05** – Caso Especial : Raiz Quadrada

**Especificação da Integração Fluxo –BD**

Pretende-se especificar e planear toda a interação entre a Base de Dados e o fluxograma.

Estamos a assumir que todo o tipo de informação relevante ao fluxograma vai ser guardada na base de dados.

**INTBD.00 –** Interacção Base de Dados/Fluxograma

**FLUX.00.01** – As formas devem ser distintas entre si.



Figura 1 – Exemplo das formas

**FLUX.00.02** – Todas as formas devem ser manipuláveis graficamente, recorrendo ao uso de pontos extensíveis, arrasto com o rato.

**FLUX.00.02.01** – O duplo clique de rato serve para inicializar a introdução de dados.



Figura 2 – Pontos de extensão

**FLUX.00.03** – As seguintes formas devem suportar multi-texto:

* Leitura
* Processo/Funções
* Escrita

**FLUX.00.04** – As linguagens de programação diferem entre si e por isso os símbolos reservados são diferentes entre as mesmas, um utilizador deve poder escolher os símbolos com que está mais familiarizado.

**FLUX.00.04.01** – Os símbolos padrão são aqueles utilizados no Portugol.

**FLUX.01.01** - O início e o fim são definidos pela mesma forma:



Figura 3 – forma para início e fim.

**FLUX.01.02** - Deve ser possível atribuir apenas uma ligação de entrada.

**FLUX.01.03** - Deve ser possível atribuir apenas uma ligação de saída.

**FLUX.01.04** – O texto define o início ou fim e por isso deve estar protegido e não pode ser alterado.

**FLUX.01.04.01** – O duplo clique de rato altera o sentido da forma.

**FLUX.02.01** - A união reúne o fluxo do programa.



Figura 4 - União

**FLUX.02.02** - Deve ser possível atribuir uma ou mais ligações de entrada.

**FLUX.02.03** - Deve ser possível atribuir apenas uma ligação de saída.

**FLUX.03.01** – A leitura define a declaração de variáveis.



**FLUX.03.01.01 –** Podem ser definidas várias variáveis separadas pelo símbolo “,”;

Figura 5 - Leitura

**FLUX.03.02** - O tipo de variáveis, vão ser deduzidas pelo código.

**FLUX.03.03** - As variáveis podem ser Inteiros, Reais, Booleans, Strings e Char.

**FLUX.03.04** - O valor da inicialização da variável tem de ser compatível com o tipo de variável definido.

**FLUX.03.05** - O valor da inicialização pode ser omitido, logo ela é inicializada por defeito.

**FLUX.03.06** - É possível omitir a palavra variável.

**FLUX.03.07** - Se a variável contém uma String, a mesma, começa por uma letra ou caracter e não pode ser nenhuma palavra reservada da linguagem.

**FLUX.03.08** – Podem ser definidos arrays vazios seguindo a forma:

“Nome\_array[tamanho\_array]”

**FLUX.03.09** - A forma da declaração da variável permite uma entrada de informação e uma saída de informação;

**FLUX.04.01** - O mesmo símbolo representa expressões e/ou funções.



Figura 6 – Processo/Funções



**FLUX.04.02** - No mesmo bloco podem ser executadas várias tarefas.

Figura 7 – Esquema de introdução de dados

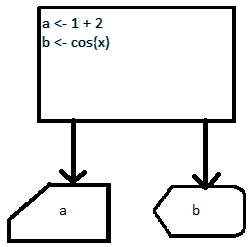
**FLUX.04.03** - A forma tem suporte para multi-texto.

**FLUX.04.04** - Pode ter vários pontos de entrada mas apenas tem uma entrada válida. Pode ter uma ou mais saídas.



Figura 8 – Entradas e Saídas

**FLUX.04.05** - As saídas possíveis são, a decisão, o return e a escrita.



**FLUX.04.06** - Deve ser possível obter resultados individuais.

Figura 9 – Saída de dados separadamente

**FLUX.04.07** - Devem estar reservados símbolos para operações aritméticas.

**FLUX.04.08** - Soma ' + '

Subtração ' - '

Divisão ' / '

Multiplicação ' \* '

Percentagem ' % '

Potência ' \*\* '

**FLUX.04.09** - As funções devem ser definidas com o nome da função seguido dos seus parâmetros entre parêntesis.

**FLUX.04.10** - Uma função pode ter um ou mais parâmetros e os mesmos devem estar separados por vírgulas, exemplo: cos(x).

**FLUX.04.11** - Deve ser possível a definição de arrays e a atribuição de valores a arrays vazios.



Figura 10 – Exemplo de inicialização de um array

**FLUX.04.11.01** - O sistema deve permitir que a inicialização do array seja feita com dados compreendidos entre chavetas e separados por “,”.

**FLUX.04.11.02** – Se as posições não estiverem todas ocupadas, são inicializadas a “0”.

**FLUX.05.01** – Decisão permite a criação de ciclos, comparações e decisões.



Figura 11 – Decisão

**FLUX.05.02** - Deve ser possível atribuir apenas uma conexão de entrada.

**FLUX.05.03** - Deve ser possível atribuir, no mínimo, uma conexão de saída e, no máximo, dois conexões de saída.

**FLUX.05.04** - Ao selecionar a primeira atribuição de saída, deve-lhe ser atribuída por defeito a condição “Sim”.

**FLUX.05.05** - Ao selecionar a segunda atribuição de saída, deve-lhe ser atribuída por defeito a condição “Não”.

Figura 12 – Relação conectores/decisão

**FLUX.05.05** - Dentro da forma deve ser permitida a escrita de operadores relacionais e lógicos.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Operadores Relacionais: | < | > | = | != | <= | >= |

|  |  |  |  |
| --- | --- | --- | --- |
| Operadores Lógicos: | E | OU | NAO |
| Operadores Lógicos(Alternativo): | && | || | ! |

**FLUX.05.06** - Um ciclo deve ser representado na seguinte forma:

Figura 13 – Ciclo

**FLUX.06.01** – A escrita permite o retorno de dados/resultados de operações simples.



Figura 14 - Escrita

**FLUX.06.02** - Possui uma só entrada e saída de conetores.

**FLUX.06.03** - É responsável pela escrita do valor de variáveis, funções, texto ou expressões.

**FLUX.06.04** - Permite a escrita de um ou mais elementos, dos quais, variáveis, funções, textos ou expressões, separadas por uma vírgula “,”.

**FLUX.06.05** – A escrita processa-se pela ordem indicada no “fluxograma” ou no código (Ex: a,b,c ; imprime a, seguido de b e for fim c).

**FLUX.06.06** - Para a escrita variáveis é escrito o valor correspondente dessa variável.

**FLUX.06.07** - Para escrita de texto, é mostrado o texto colocado entre aspas(“”) (Ex: “123” imprime 123).

**FLUX.06.08** - Para o caso da escrita de uma função esta é caracterizada pelo valor de retorno, escrevendo esse valor.

**FLUX.06.09** - Esta forma suporta caracteres reservados, tais como o “\n” para mudança de linha (enter), “\t” para a tabulação e \”para as aspas(“).

**FLUX.07.01** – o return representa o resultado de uma função



Figura 15 - return

**FLUX.07.02** - Dentro do símbolo de retorno é permitido retornar:

* Vários tipos normais de variáveis (char, int, reais, bolean, strings);
* Números;
* Expressões matemáticas (cos, sin, tan, etc);
* Operações matemáticas;

**FLUX.07.03** - Deve ser possível atribuir apenas uma conexão de entrada e uma saída

**FLUX.07.04** - Não é obrigatório a inclusão de retorno a não ser que a função o especifique (não ser do tipo void ou conter parâmetros)



Figura 16 – Resultado de uma função

**FLUX.08.01** – Os conectores ligam as formas entre si, são unidirecionais e podem estabelecer condições de saída para a forma de decisão como o especificado em “**FLUX.05”.**

Figura 17 – Exemplo de conector

**Especificação da forma Switch Case**

Figura 18 –SWITCH CASE

**FLUX.09.00** – A forma Switch é, essencialmente, um conjunto de ciclos *if* encadeados

**FLUX.09.01** – Tem somente uma conexão de entrada.

**FLUX.09.02** – Tem obrigatoriamente uma conexão de saída, proveniente da forma União, sendo que entre o Switch e a União pode ter várias conexões tanto de entrada como de saída.

Figura 19 – Exemplo em como entre o Switch e a União pode existir várias conexões.

**FLUX.09.03** – Cada conexão entre a condição e o caso, deve conter uma label editável que especifica de que o caso se trata (Ex: 1,2,3;Janeiro,Fevereiro,Março)



Figura 20 – Label editável na forma do Switch Case

**FLUX.09.04** - Dentro da forma deve ser permitida a escrita de operadores relacionais e lógicos

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Operadores Relacionais: | < | > | = | != | <= | >= |

|  |  |  |  |
| --- | --- | --- | --- |
| Operadores Lógicos: | E | OU | NAO |
| Operadores Lógicos(Alternativo): | && | || | ! |

**FLUX.09.05** – A condição deve ser especificada na forma Switch mas o resultado deve ser especificado em cada caso.



Figura 21 – A condição que é especificada no Switch Case

**Especificação da forma FOR**



**FLUX.10.00 –** Esta forma, representa na sua essência o ciclo FOR, e tem o seguinte formato:

**FLUX.10.01 –** Tal como o próprio ciclo FOR, esta forma está dividida em 3 secções: definição(i=0), condição(i<=100) e o incremento(i++).

Exemplo:



**FLUX.10.02 -** A forma FOR dispõe de duas conexões de entrada: uma proveniente de outras formas e outra da instrução que executa. As conexões de entrada apenas podem ser encaixadas na secção de definição.

A forma suporta tantas condições de entrada quanto o número de instruções.

**FLUX.10.03 -** A forma FOR dispõe de duas conexões de saída: uma que saí do ciclo para outras formas e outra que indica que instruções o ciclo vai executar.

As conexões de saída apenas podem ser encaixadas na secção de incremento. A forma suporta tantas condições de saída quanto o número de instruções.



**FLUX.10.04** - Dentro da secção condição deve ser permitida a escrita de operadores relacionais e lógicos

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Operadores Relacionais: | < | > | = | != | <= | >= |

|  |  |  |  |
| --- | --- | --- | --- |
| Operadores Lógicos: | E | OU | NAO |
| Operadores Lógicos(Alternativo): | && | || | ! |

Especificação de Comentários

**FLUX.11.01 -** O sistema deve permitir a introdução de comentários mediante a forma:

Figura 1 - Forma escrita

**FLUX.11.02 -** O sistema deve permitir a introdução de texto dentro da forma comentário:

**FLUX.11.03 -** O comentário pode estar ao lado do grafo ou associado a um nó:

Inicio

Fim

Ola Mundo

Aqui imprime o Ola Mundo

Figura 3 - Comentário ao lado do node escrita

Inicio

Fim

Ola Mundo

Aqui imprime o Ola Mundo

Figura 4 - Comentário com nó de escrita associado

**FLUX.11.04 -** A associação de um comentário a um nó é realizada com uma seta a tracejado.

Ola Mundo

Aqui imprime o Ola Mundo

Figura 5 - Comentário com seta a tracejado a associar à escrita

Especificação editor de expressões

**FLUX.12.01** - O sistema deve permitir ao utilizador um editor de expressões.

**FLUX.12.02** - O editor de expressões deve incluir uma secção de introdução de expressões e operadores.

**FLUX.12.03** - O sistema deve conter um conjunto de operadores lógicos suportados, que na escolha de um deles inclui esse operador à secção de introdução de expressões e operadores.

**FLUX.12.04** - O sistema deve disponibilizar uma listagem com as funções existentes criadas pelo utilizador.

**FLUX.12.05** - O sistema deve disponibilizar uma listagem com funções pré definidas categorizadas por tipo de funcionalidade (ex: Funções matemáticas, Operações numéricas, Edição de Strings).

**FLUX.12.06** - O sistema deve apresentar uma descrição da função, o tipo de valor retornado e possíveis inputs após escolher uma função pré definida.

**FLUX.12.07** - O sistema deve permitir a pesquisa de uma função existente no sistema.

Layout



Interacção da página de perguntas com o IDE

**FLUX.13.01 –** O teste é iniciado com a disponibilização de um menu designado de menu enúnciado.

**FLUX.13.02 –** Este menu conter as perguntas do teste juntamente com a sua descrição.

**FLUX.13.03 -** A página de testes, deve conter um botão, designado de “Iniciar Teste” para interagir com uma versão modificada para testes do Fluxograma IDE.

**FLUX.13.04 –** Ao clicar em “Iniciar Teste”, o sistema deve iniciar o ambiente IDE com o conteúdo dessa pergunta.

**FLUX.13.05 –** O menu deve dispor de um mecanismo que permita averiguar se a questão foi respondida ou não.

**FLUX.13.06 –** É a partir do menu que deve ser efectuada a selecção da linguagem (ex: Português, Inglês) a que vai decorrer o teste.

**FLUX.13.07 –** O ambiente IDE contém um separador separadores: horizontal.

**FLUX.13.08 –** O separador horizontal refere-se a todos os exercícios do teste.

**FLUX.13.09 –** Para efetuar a navegação entre as perguntas, o utilizador deve clicar no separador horizontal que contem a pergunta que pretende responder.

**FLUX.13.10 –** Ao passar o rato por cima do separador da pergunta actual, deve aparecer uma *tooltip*(notificação) que contém o enunciado da pergunta.

**FLUX.13.11 –** Durante a execução do teste deve constar um cronómetro que dispõe o tempo decorrido desde o início do teste.

**FLUX.13.11.01–** Assim que o cronómetro chega ao tempo limite estipulado, o teste conclui automáticamente e deve gravar automaticamente as respostas inseridas.

**FLUX.13.12 –** O sistema deve permitir um mecanismo que guarde o exercício individual que o utilizador resolveu e que guarda o conjunto de todos os exercícios resolvidos.

**FLUX.13.12.01 -** O fluxograma IDE deve conter um botão para submeter apenas a pergunta que o utilizador realizou.

**FLUX.13.12.02 -** O fluxograma IDE deve conter um botão para submeter todas as perguntas realizadas.

**FLUX.13.13 –** A partir do momento que o cronómetro alcança o tempo limite do teste, o menu enunciado é disponibilizado.

**FLUX.13.14 –** A partir deste menu, deve ser possível “Entregar o teste”, ou seja, guardar os dados das respostas inseridas permanentemente.

**FLUX.13.15 -** Mediante a entrega do teste, o sistema deve permitir a correcção desse teste.

**FLUX.13.16 –** A correcção do teste implica dois cenários:

* A integração do sistema tutor, que deverá ajudar o utilizador a corrigir os erros efectuados no teste por si próprio.
* Deverá ser disponibilizada a solução dos exercícios caso não tenha sido embutido uma análise do sistema tutor para esse determinado exercício. (Não há interacção entre o tutor e o utilizador em determinado exercício).

Especificação de Funções

**FLUX.14.00 –** Uma função deve ser inicializada com a forma “inicio”.

**FLUX.14.01 –** Quando se trata de uma função com parâmetros a forma “inicio” deve poder ser alterado para receber os mesmos.

**FLUX.14.02 –** Para alterar a forma “inicio” deve ser usado o editor de expressões.

**FLUX.14.03 –** No editor deve existir o campo para função e seus parâmetros.

**FLUX.14.04 –** O output da função é a forma “return”.

**FLUX.14.05 –** Mesmo sendo o output o “return” deve ser usada a forma “fim” para terminar o programa.

**FLUX.14.06 –** Uma função criada pode ser guardada num separador/submenu à parte.

**FLUX.14.07 –** Para cada nova função criada, devem ser utilizados outros separadores/submenu.

Exemplo (no separador):







**Especificação do Parser**

**PARS.00.01** – Os erros detetados devem ser devolvidos para o ecrã do IDE.

**PARS.00.01.01**- Deve surgir uma mensagem de erro no IDE;

**PARS.00.01.02** – A mensagem deve ter um espaço próprio onde pode ser detalhada e explorada;

**PARS.00.01.03** – A mensagem deve conter o tipo de erro e a linha onde este se encontra;

**PARS.00.01.04** – Ao clicar na linha do erro deve ser possível ao IDE apresentar onde o mesmo se situa;

**PARS.00.01.05** – Quando a linha em questão é mostrada, deve estar sublinhada a vermelho;

**PARS.00.01.06** – A mensagem de erro deve ser acompanhada de uma recomendação que ao ser clicada em certos casos pode corrigir automaticamente o problema;

**PARS.01.01** - Símbolo indefinido: Refere para um caracter/palavra que não está definido.

**PARS.01.01.01** – Recomendação: Verificar a ortografia;

**PARS.01.02** – Opcode ou pseudo-op indefinidas.

**PARS.01.02.01** – Recomendação: Verificar disponibilidade da operação;

**PARS.01.03** – Modo de endereçamento não disponível.

**PARS.01.03.01** – Recomendação: Verificar modos de endereçamentos correspondentes à operação;

**PARS.01.04** – Erro de expressão.

**PARS.01.04.01 –** Recomendação: Verificar parêntesis ou iniciar com uma expressão mais simples;

**PARS.01.05** – Erro de phasing: ocorre quando o valor de uma variável muda de um passo para outro.

**PARS.01.05.01 –** Recomendação: Remover símbolos indefinidos e de seguida remover referências a esses mesmos símbolos;

**PARS.01.06** – Erro de endereço.

**PARS.01.06.01** – Recomendação: Verificar memória disponível;

**PARS.02.01** – Erros de sintaxe: erros de gramática e pontuação.

**PARS.02.01.01** – Recomendação: Verificar ortografia;

**PARS.02.02** – Erros de runtime: aparecem à medida que o programa é executado.

**PARS.02.02.01** – Recomendação: Rever o código, por exemplo, procurar divisões por zero;

**PARS.02.03** – Erros lógicos: erros básicos em algoritmos ou procedimentos, apenas detetados quando o resultado destes é anómalo.

**PARS.02.03.01** – Recomendação: Usar mecanismos de debug (passo a passo);

**PARS.02.04** – Erros de threading: Ocorrem quando se usam threads, exemplos podem ser deadlocks e concorrência no acesso.

**PARS.02.04.01** – Recomendação: Utilizar mecanismos de defesa, por exemplo, semáforos;

**PARS.02.05** – Exceções: Não são erros fatais mas podem provocar problemas na execução do programa.

**PARS.02.05.01** – Recomendação: Utilizar técnicas de controlo de exceções, por exemplo, “try catch”;

**PARS.03.01** – Erros de ligação: Quando uma forma do fluxograma não está ligada ou faltam ligações.

**PARS.03.01.01** – Recomendação: Procurar formas não ligadas;

**PARS.03.02** – Erros de código: Quando na tradução de código para fluxograma, os erros do anterior se transferem para o diagrama.

**PARS.03.02.01** – Recomendação: Corrigir erro no código através da forma que o contém;

**PARS.03.03** – Sem início ou fim: Quando estão em falta as formas de início e fim do programa.

**PARS.03.03.01** – Recomendação: Repor a forma(s) que estiver(em) em falta.

**PARS.04.00 -** Uma string é uma cadeia de carateres ASCII delimitados por aspas, que são armazenadas em "instâncias" da classe String

String ola = “Ola Mundo!”

**PARS.04.01 -** O parser deve estar preparado para percorrer uma sequência de caracteres.

**PARS.04.02 -** O parser deve estar preparado para interpretar o carater “+”(ou a vírgula 🡪 **para decidir**) como separador de strings.

// concatenação (\n = pula linha)  
String aula = ola **+** "\n " **+** mundo;

**PARS.04.03 -** O parser deve estar preparado para interpretar separadores como /n, /t, etc.

**PARS.04.04 -** O parser estar preparado para detectar a comparação de strings

**PARS.04.05 -** O parser estar preparado para detectar conversões valores de tipos nativos para strings e vice-versa

**PARS.05.00 -** Para definir uma raiz quadrada a melhor forma seria através de uma função **sqrt()**, pois seria mais facilmente associada a outras linguagens que recorram a bibliotecas matemáticas com esta mesma notação.

Para a raíz quadrada teríamos então:

**Sqrt(**parâmetro **)**

**PARS.05.01 -** Para as **n raízes** (cúbica, à quarta, etc..) será necessário criar uma função custom **sqrt()** que recebe-se no máximo 2 parâmetros onde o primeiro seria o valor a calcular e o segundo o valor do radical, onde poderíamos recorrem ao **math.pow** determinar as n raizes. Caso não seja indicado nenhum valor no radical seria utilizada a função acima, com o primeiro parâmetro na raiz.

**root(** param1**,**param2 **)**

**Especificação da Integração Fluxo –BD**

**INTBD.00.00 –** O sistema deve guardar todas as informações necessárias relativamente às posições, condições e conexões de cada forma.

(Isto deve ser feito de modo a que, sempre que o utilizador reinicie a sessão, todo o ambiente gráfico permaneça inalterado relativamente à sessão anterior)

**INTBD.00.01 –** Para cada forma devem ser guardadas todas instruções e condições inseridas nessa forma assim que o utilizador sai da sessão.

**INTBD.00.02 –** As instruções e condições acima referidas deverão ser recarregadas numa sessão seguinte.

**INTBD.00.03-** Mediante o fecho de sessão, as coordenadas das formas e conexões inseridas devem ser guardadas

**INTBD.00.04 –** As formas devem aparecer na mesma posição quando um utilizador reinicia a sessão

**INTBD.00.05 –**Para cada conexão, deve ser guardado dois tipos de informações:

* De que forma veio essa conexão
* Em que forma a conexão se encontra acoplada (a parte que tem a seta!)

**INTBD.00.06 –** O sistema deve perguntar se o utilizador pretende guardar as alterações quando este tentar finalizar a sessão.

**INTBD.00.07 –** O sistema deve permitir ao utilizador guardar as alterações feitas no fluxograma.

**INTBD.00.08 –** O sistema deve armazenar a informação contida no fluxograma a cada 5 ou 10 minutos se houver alguma alteração no fluxograma

**INTBD.00.09 –** O sistema deve permitir o restauro do estado do fluxograma(ou do último ponto de restauro) se a sessão anterior foi encerrada de forma abrupta.

**INTBD.00.10 –** Se não for possível o restauro do estado do fluxograma acima referido, então o sistema deve permitir ao utilizador continuação do trabalho num ponto de restauro anterior.

**INTBD.00.11 –** O sistema deve ter pelo menos 2 pontos de restauro para o requesito acima referido