Linguagens e Ambientes de Programação

Exame de Época Normal 2007/2008 – Proposta de Resolução

1. **F** Os sistemas de tipos das linguagens com tipificação estática, como é o caso do C++, são conservadores. Para garantir a segurança, o compilador assume sempre o pior caso, mesmo que este nem sempre ocorra, i.e. por vezes o compilador poderá emitir um erro de compilação, o que não quer dizer que em tempo de execução haja qualquer problema.

   F Todos os problemas resolúveis por computador podem ser programados usando apenas a parte funcional do ML. Do ponto de vista do poder computacional, todas as linguagens normais são equivalentes.

   V Em ML existe aplicação parcial e em C++ parâmetros por defeito.

   F, **V**, **F**, **V**, **V**

   F As fases da compilação são:
   1. Leitura de caracteres do ficheiro fonte;
   2. Análise lexical;
   3. Análise sintáctica;
   4. Análise semântica (verificação de tipos);
   5. Melhoramento e completação da árvore abstracta;
   6. Geração de código intermédio;
   7. Optimização de código intermédio;
   8. Geração de código máquina.

   Para se fazer a verificação do tipo de uma expressão, a estrutura da expressão tem de ser conhecida. Como tal, a análise sintáctica vem antes da verificação de tipos.

   F Um programa compilado não precisa do compilador visto que todos os dados estão carregados no ficheiro objecto.

2.

1. A função `f` recebe dois argumentos, sendo que o primeiro é uma função.

   ```
   f : (__ -> __) -> __ -> __
   ```

2. Sabendo que a função `fst` recebe `x` como argumento e que o seu tipo é `'a * 'b -> 'a`, pode-se admitir que `x` é um par ordenado `'a * 'b`.

   ```
   f : (__ -> __) -> 'a * 'b -> __
   ```

3. A função `g` recebe o resultado da função `fst x` como argumento, que se sabe ser `'a`. O tipo de retorno é desconhecido, pelo que se considera um tipo qualquer.

   ```
   f : ('a -> 'c) -> 'a * 'b -> __
   ```

4. O retorno da função `f` irá ser o resultado da função `g`.

Assim sendo:

```
f : ('a -> 'c) -> 'a * 'b -> 'c
```

3.

a)
```
let rec prefix t =
    match t with
        | Entity e -> [e]
        | Question(q,n,y) -> q::(prefix n @ prefix y)
;;
```

b)
```
let xiferp l = let (t,[]) = xiferpAux l in t ;;

let rec xiferpAux l =
    match l with
        | x::xs ->
            if (not(isQuestion x)) then (Entity x,xs)
            else let (l,resto1) = xiferpAux xs in
                let (r,resto2) = xiferpAux resto1 in
                    (Question(x,l,r),resto2)
;;
```

4.

a) Neste caso em particular, a utilização de escopo dinâmico não alteraria o comportamento do programa, dado que a variável `i` foi declarada localmente.

b)

	PC	?	-- X --
25:	SL	00	
	DL	19	
	b	4	
	a	18	
	PC	?	-- Z --
20:	SL	14	
	DL	14	
	b	7	
	a	4	
	PC	?	-- Y --
15:	SL	09	
	DL	09	
	b	3	
	a	4	
	PC	?	-- X --
10:	SL	00	
	DL	04	
	b	3	
	a	3	
	PC	?	-- main --
05:	SL	00	
	DL	00	
	i	3 -> 4	-- start --
	PC	?	
	SL	?	
00:	DL	?	

5. ***.h**

```
class Vector {
    protected:
        int size;
        int used;
    public:
        Vector(int size);
        virtual ~Vector();
        virtual double Get(int i) const = 0;
        virtual void Set(int i, double v) = 0;
        virtual int Find(double v) const = 0;
};

class NormalVector : public Vector {
    protected:
        double *vector ;
    public:
        NormalVector(int size);
        virtual ~NormalVector();
        double Get(int i) const;
        void Set(int i, double v);
        int Find(double v) const;
};

class SparseVector : public Vector {
    protected:
        int *indexes;
        double *values;
    public:
        SparseVector(int size);
        virtual ~SparseVector();
        double Get(int i) const;
        void Set(int i, double v);
        int Find(double v) const;
};

class CleverVector : public Vector {
    protected:
        Vector* vector;
        int zeros;
        bool normal;
    public:
        CleverVector(int size);
        virtual ~CleverVector();
        double Get(int i) const;
        void Set(int i, double v);
        int Find(double v) const;
        void toNormal();
        void toSparse();
};
```

***.cpp**

```cpp
#include "Vector.h"
#include<iostream>
#include<string>

using namespace std;

Vector::Vector(int size) : size(size), used(0) {}
Vector::~Vector() {}

NormalVector::NormalVector(int size) : Vector(size) {
    vector = new double[size];
}
NormalVector::~NormalVector(){}

double NormalVector::Get(int i) const {
    return vector[i];
}

void NormalVector::Set(int i, double v) {
    vector[i] = v;
    used++;
}

int NormalVector::Find(double v) const {
    for(int i = 0; i < size; i++)
        if(v == vector[i])
            return i;

    return -1;
}

SparseVector::SparseVector(int size) :
Vector(size), values(new double[size]), indexes(new int[size]) {}
SparseVector::~SparseVector() {}

double SparseVector::Get(int i) const {
    for(int j = 0; j < used; j++)
        if(indexes[j] == i)
            return values[j];

    return 0;
}
void SparseVector::Set(int i, double v) {
    int pos = Find(i);

    if(pos == -1) {
        indexes[used] = i;
        values[used] = v;
        used++;
    }
    else values[pos] = v;
}

int SparseVector::Find(double v) const {
    for(int i = 0; i < used; i++)
        if(values[i] == v) return indexes[i];

    return -1 ;
}
```

```
CleverVector::CleverVector(int size) :
    Vector(size), vector(new NormalVector(size)), normal(true) {}
CleverVector::~CleverVector(){}

double CleverVector::Get(int i) const {
    return vector->Get(i);
}

void CleverVector::Set(int i, double v) {
    if((v == 0) && (Get(i) != v))
        zeros++;

    vector->Set(i, v);

    if(((zeros/used) > 0.5) && normal)
        toSparse();

    else if(((zeros/used) < 0.5) && !normal)
        toNormal();
}

int CleverVector::Find(double v) const {
    vector->Find(v);
}

void CleverVector::toNormal() {
    Vector *vtemp = new NormalVector(size);

    for(int i = 0; i < size; i++)
        vtemp->Set(i, Get(i));

    normal = true;
    vector = vtemp;
}

void CleverVector::toSparse() {
    Vector *vtemp = new SparseVector(size);

    for(int i = 0; i < size; i++)
        vtemp->Set(i, Get(i));

    normal = false;
    vector = vtemp;
}
```