CC41A Lenguajes de Programación

Control 1 - Otoño 2006

Profesor: Luis Mateu

Pregunta 1

Se requiere calcular el tamaño del registro de activación de un procedimiento para el caso de un lenguaje de programación como Java. Se decide simplificar el problema recurriendo a un lenguaje minimal que posee instrucciones para declarar variables y agrupar instrucciones. Los siguientes son ejemplos de programas válidos y cómo se calcula el tamaño del registro de activación para cada uno de ellos:

`int i; double x;`	4 bytes para i + 8 bytes para x. Total: 12 bytes.

`int i; double x;`
`{ int j; }`
`{ double y; }`
	4 bytes para i+ 8 bytes para x +
	max(4 bytes para j, 8 bytes para y).
	Total: 20 bytes.

`int i;`
`{ int j; }`
`int k;`
`{ double x;`
`  { int l; double y; }`
`  { double z; }`
`}`
	4 bytes para i + 4 bytes para k+
	max(4 bytes para j, 8 bytes para x +
	max(4 bytes para l + 8 bytes para y, 8 bytes para z)).
	Total: 28 bytes.

(a) Escriba una especificación para CUP que genere un analizador sintáctico que acepte los 3 ejemplos anteriores. *No* escriba el analizador léxico ni incluya instrucciones que no aparecen en los ejemplos.

(b) Modifique su especificación asociando acciones que calculen el tamaño requerido para el registro de activación. Hágalo *sin* construir un árbol de derivación o de sintaxis abstracta. Indicación: para cada grupo de instrucciones, calcule (i) el espacio requerido por las variables declaradas directamente en ese bloque, y (ii) el máximo espacio requerido para variables declaradas en bloques internos.

Pregunta 2

i. Para el siguiente programa en C, haga 3 diagramas de variables y objetos conceptualmente vivos después de ejecutar las instrucciones en (A), (B) y (C). Indique para cada variable u objeto su nombre (si lo hay), localidad (espacio en memoria), tipo y contenido (con sus campos). Desde un punto de vista de la implementación señale también si se ubican en la pila, en el *heap* u otro sitio.

```
struct N { int x; struct N *l, *r; } root= { 0, NULL, NULL};
int main() {
  struct N *left=(struct N*)malloc(sizeof(struct N));
  left->x= 0; left->l= left->r= NULL;
  root.l= left;
  { struct N right;
    right.l= right.r= NULL;
    root.r= &right; /*  (A)  */
  }
  /* (B) */
  { int z;
    root.r= (struct N*)&z; /* (C) */
} }
```

ii. Suponga que en (A) se escriben la siguientes expresiones:

```
root.l->r->x
(&right)[5].x
```

Construya árboles de sintaxis abstracta para cada una de las expresiones, etiquetando todos los nodos con el tipo de la sub-expresión a la cual se encuentra asociado.