BÁO CÁO THỰC HÀNH XÂY DỰNG CHƯƠNG TRÌNH DỊCH

GIẢNG VIÊN HƯỚNG DẪN: THẦY TRẦN VĨNH ĐỨC

BÁO CÁO SEMANTIC -DAY 1

HỌ VÀ TÊN SINH VIÊN: NGUYỄN VĂN QUÝ

MÃ SỐ SINH VIÊN: 20210729

MÃ LỚP THỰC HÀNH: 151933

Task: hãy lập trình cho những hàm được đánh dấu TODO

1. Type* duplicateType(Type* type)

```
43  Type* duplicateType(Type* type) {
44      // TODO
45      Type* dupType = (Type*) malloc(sizeof(Type));
46      dupType->typeClass = type->typeClass;
47      dupType->arraySize = type->arraySize;
48      dupType->elementType = type->elementType;
49      return dupType;
50  }
```

- Ý nghĩa: trả về bản sao của kiểu đã thiết kế
- Cách thức:
    - Tạo đối tượng Type mới có tên dupType
    - Lần lượt ánh xạ các thuộc tính của dupType với tham số type được truyền vào từ hàm: typeClass, arraySize, elementType
    - Trả về dupType
2. int compareType(Type* type1, Type* type2)

```
52  int compareType(Type* type1, Type* type2) {
53      // TODO
54      if (type1->typeClass != type2->typeClass){
55          return 0;
56      }else if(type1->arraySize != type2->arraySize){
57          return 0;
58      }else if(type1->arraySize != 0 &&type1->elementType->typeClass != type2->elementType->typeClass){
59          return 0;
60      }else{
61          return 1;
62      }
63  }
```

- Ý nghĩa:
    - So sánh hai đối tượng Type để kiểm tra xem chúng có cùng loại hay không.
- Cách thức hàm làm nhiệm vụ đó:
    - Kiểm tra thuộc tính typeClass của hai đối tượng Type. Nếu khác nhau, trả về 0.
    - Kiểm tra kích thước mảng (arraySize) nếu typeClass là mảng. Nếu khác nhau, trả về 0.
    - Nếu cả hai là mảng, so sánh kiểu của các phần tử mảng thông qua elementType. Nếu khác, trả về 0.
    - Nếu tất cả điều kiện đều thỏa mãn, trả về 1 để biểu thị rằng hai đối tượng Type tương đồng.
3. void freeType(Type* type)

```
void freeType(Type* type) {
  // TODO - free elementType them type
  if (type->arraySize != 0){
    free(type->elementType);
  }
  free (type);
}
```

- Ý nghĩa:
  - Giải phóng bộ nhớ được cấp phát cho một đối tượng Type.
- Cách thức hàm làm nhiệm vụ đó:
  - Nếu kiểu dữ liệu là mảng (arraySize != 0), giải phóng bộ nhớ của elementType trước.
  - Giải phóng chính đối tượng Type.

4. ConstantValue* makeIntConstant(int i)

```
ConstantValue* makeIntConstant(int i) {
  // TODO
  ConstantValue* constant = (ConstantValue*)malloc(sizeof(ConstantValue));
  constant->type = TP_INT;
  constant->intValue = i;
  return constant;
}
```

- Ý nghĩa:
  - Tạo một hằng số nguyên (ConstantValue) và khởi tạo giá trị của nó.
- Cách thức hàm làm nhiệm vụ đó:
  - Cấp phát bộ nhớ cho một đối tượng ConstantValue.
  - Gán loại (type) là TP_INT.
  - Gán giá trị nguyên (intValue) bằng tham số đầu vào.
  - Trả về con trỏ đến đối tượng vừa được tạo.

5. ConstantValue* makeCharConstant(char ch)

```
ConstantValue* makeCharConstant(char ch) {
  // TODO
  ConstantValue* constant = (ConstantValue*)malloc(sizeof(ConstantValue));
  constant->type = TP_CHAR;
  constant->charValue = ch;
  return constant;
}
```

- Ý nghĩa:
  - Tạo một hằng số nguyên (ConstantValue) và khởi tạo giá trị của nó.
- Cách thức hàm làm nhiệm vụ đó:
  - Cấp phát bộ nhớ cho một đối tượng ConstantValue.
  - Gán loại (type) là TP_INT.
  - Gán giá trị nguyên (intValue) bằng tham số đầu vào.
  - Trả về con trỏ đến đối tượng vừa được tạo.

6. ConstantValue* duplicateConstantValue(ConstantValue* v)

```
91  ConstantValue* duplicateConstantValue(ConstantValue* v) {
92    // TODO
93    ConstantValue* dupConstant = (ConstantValue*)malloc(sizeof(ConstantValue));
94    dupConstant->type = v->type;
95    if (v->type == TP_INT){
96      dupConstant->intValue = v->intValue;
97    } else {
98      dupConstant->charValue = v->charValue;
99    }
100   return dupConstant;
101 }
```

- Ý nghĩa:
    - Tạo một hằng số nguyên (ConstantValue) và khởi tạo giá trị của nó.
- Cách thức hàm làm nhiệm vụ đó:
    - Cấp phát bộ nhớ cho một đối tượng ConstantValue.
    - Gán loại (type) là TP_INT.
    - Gán giá trị nguyên (intValue) bằng tham số đầu vào.
    - Trả về con trỏ đến đối tượng vừa được tạo.

7. Object* createConstantObject(char *name)

```
125 Object* createConstantObject(char *name) {
126   // TODO
127   Object* program = (Object*) malloc(sizeof(Object));
128   strcpy(program->name, name);
129   program->kind = OBJ_PROGRAM;
130   program->progAttrs = (ProgramAttributes*) malloc(sizeof(ProgramAttributes));
131   program->progAttrs->scope = createScope(program,NULL);
132   symtab->program = program;
133
134   return program;
135 }
```

- Ý nghĩa:
    - Tạo một hằng số nguyên (ConstantValue) và khởi tạo giá trị của nó.
- Cách thức hàm làm nhiệm vụ đó:
    - Cấp phát bộ nhớ cho một đối tượng ConstantValue.
    - Gán loại (type) là TP_INT.
    - Gán giá trị nguyên (intValue) bằng tham số đầu vào.
    - Trả về con trỏ đến đối tượng vừa được tạo.

8. Object* createTypeObject(char *name)

```
137 Object* createTypeObject(char *name) {
138   // TODO
139   Object* typeObject = (Object*) malloc(sizeof(Object));
140   strcpy(typeObject->name, name);
141   typeObject->kind = OBJ_TYPE;
142   typeObject->typeAttrs = (TypeAttributes*)malloc(sizeof(TypeAttributes));
143   typeObject->typeAttrs->actualType = (Type*)malloc(sizeof(Type));
144   return typeObject;
145 }
```

- Ý nghĩa:
    - Tạo một hằng số nguyên (ConstantValue) và khởi tạo giá trị của nó.
- Cách thức hàm làm nhiệm vụ đó:
    - Cấp phát bộ nhớ cho một đối tượng ConstantValue.
    - Gán loại (type) là TP_INT.
    - Gán giá trị nguyên (intValue) bằng tham số đầu vào.

- Trả về con trỏ đến đối tượng vừa được tạo.

9. Object* createVariableObject(char *name)

```
Object* createVariableObject(char *name) {
  // TODO
  Object* variableObject = (Object*) malloc(sizeof(Object));
  strcpy(variableObject->name, name);
  variableObject->kind = OBJ_VARIABLE;
  variableObject->varAttrs = (VariableAttributes*)malloc(sizeof(VariableAttributes));
  variableObject->varAttrs->type = (Type*)malloc(sizeof(Type));
  variableObject->varAttrs->scope = createScope(variableObject, symtab->currentScope);

  return variableObject;
}
```

- Ý nghĩa:
  - Tạo một hằng số nguyên (ConstantValue) và khởi tạo giá trị của nó.
- Cách thức hàm làm nhiệm vụ đó:
  - Cấp phát bộ nhớ cho một đối tượng ConstantValue.
  - Gán loại (type) là TP_INT.
  - Gán giá trị nguyên (intValue) bằng tham số đầu vào.
  - Trả về con trỏ đến đối tượng vừa được tạo.

10. Object* createFunctionObject(char *name)

```
Object* createFunctionObject(char *name) {
  // TODO
  Object* functionObject = (Object*) malloc(sizeof(Object));
  strcpy(functionObject->name, name);
  functionObject->kind = OBJ_FUNCTION;
  functionObject->funcAttrs = (FunctionAttributes*)malloc(sizeof(FunctionAttributes));
  functionObject->funcAttrs->paramList = (ObjectNode*)malloc(sizeof(ObjectNode));
  functionObject->funcAttrs->returnType = (Type*)malloc(sizeof(Type));
  functionObject->funcAttrs->scope = createScope(functionObject, symtab->currentScope);

  return functionObject;
}
```

- Ý nghĩa:
  - Tạo một hằng số nguyên (ConstantValue) và khởi tạo giá trị của nó.
- Cách thức hàm làm nhiệm vụ đó:
  - Cấp phát bộ nhớ cho một đối tượng ConstantValue.
  - Gán loại (type) là TP_INT.
  - Gán giá trị nguyên (intValue) bằng tham số đầu vào.
  - Trả về con trỏ đến đối tượng vừa được tạo.

11. Object* createProcedureObject(char *name)

```
Object* createProcedureObject(char *name) {
  // TODO
  Object* procedureObject = (Object*) malloc(sizeof(Object));
  strcpy(procedureObject->name, name);
  procedureObject->kind = OBJ_PROCEDURE;
  procedureObject->procAttrs = (ProcedureAttributes*)malloc(sizeof(ProcedureAttributes));
  procedureObject->procAttrs->paramList = (ObjectNode*)malloc(sizeof(ObjectNode));
  procedureObject->procAttrs->scope = createScope(procedureObject, symtab->currentScope);

  return procedureObject;
}
```

- Ý nghĩa:
  - Tạo một hằng số nguyên (ConstantValue) và khởi tạo giá trị của nó.
- Cách thức hàm làm nhiệm vụ đó:

- Cấp phát bộ nhớ cho một đối tượng ConstantValue.
- Gán loại (type) là TP_INT.
- Gán giá trị nguyên (intValue) bằng tham số đầu vào.
- Trả về con trỏ đến đối tượng vừa được tạo.

12. Object* createParameterObject(char *name, enum ParamKind kind, Object* owner)

```
184  Object* createParameterObject(char *name, enum ParamKind kind, Object* owner) {
185    // TODO
186    Object* parameterObject = (Object*) malloc(sizeof(Object));
187    strcpy(parameterObject->name, name);
188    parameterObject->kind = OBJ_PARAMETER;
189    parameterObject->paramAttrs = (ParameterAttributes*)malloc(sizeof(ParameterAttributes));
190    parameterObject->paramAttrs->kind = kind;
191    parameterObject->paramAttrs->type = (Type*)malloc(sizeof(Type));
192    parameterObject->paramAttrs->function = owner;
193
194
195  }
```

- Ý nghĩa:
  - Tạo một hằng số nguyên (ConstantValue) và khởi tạo giá trị của nó.
- Cách thức hàm làm nhiệm vụ đó:
  - Cấp phát bộ nhớ cho một đối tượng ConstantValue.
  - Gán loại (type) là TP_INT.
  - Gán giá trị nguyên (intValue) bằng tham số đầu vào.
  - Trả về con trỏ đến đối tượng vừa được tạo.

13. void freeObject(Object* obj)

```
197  void freeObject(Object* obj) {
198    // TODO
199    // NOTE: thực hiện xóa các thuộc tính bên trong của object rồi
200    if (obj != NULL){
201      switch (obj->kind){
202        case OBJ_CONSTANT:
203            free(obj->constAttrs->value);
204            free(obj->constAttrs);
205            break;
206        case OBJ_VARIABLE:
207            free(obj->varAttrs->type);
208            freeScope(obj->varAttrs->scope);
209            free(obj->varAttrs);
210            break;
211        case OBJ_TYPE:
212            free(obj->typeAttrs->actualType);
213            free(obj->typeAttrs);
214            break;
```

- Ý nghĩa:
  - Tạo một hằng số nguyên (ConstantValue) và khởi tạo giá trị của nó.
- Cách thức hàm làm nhiệm vụ đó:
  - Cấp phát bộ nhớ cho một đối tượng ConstantValue.
  - Gán loại (type) là TP_INT.
  - Gán giá trị nguyên (intValue) bằng tham số đầu vào.
  - Trả về con trỏ đến đối tượng vừa được tạo.

14. void freeScope(Scope* scope)

```
void freeScope(Scope* scope) {
  // TODO
  // NOTE: thực hiện xóa lần lượt các C
  if (scope != NULL){
    freeObjectList(scope->objList);
    freeScope(scope->outer);
  }
}
```

- Ý nghĩa:
  - Tạo một hằng số nguyên (ConstantValue) và khởi tạo giá trị của nó.
- Cách thức hàm làm nhiệm vụ đó:
  - Cấp phát bộ nhớ cho một đối tượng ConstantValue.
  - Gán loại (type) là TP_INT.
  - Gán giá trị nguyên (intValue) bằng tham số đầu vào.
  - Trả về con trỏ đến đối tượng vừa được tạo.

15. void freeObjectList(ObjectNode *objList)

```
250  void freeObjectList(ObjectNode *objList) {
251    // TODO
252    // NOTE: xóa lần lượt các objectList trong obj
253    if (objList != NULL){
254        ObjectNode* temp;
255        while (objList->next != NULL){
256          temp = objList->next;
257          objList->next = temp->next;
258          free(temp);
259        }
260        free(objList);
261    }
262  }
```

- Ý nghĩa:
  - Tạo một hằng số nguyên (ConstantValue) và khởi tạo giá trị của nó.
- Cách thức hàm làm nhiệm vụ đó:
  - Cấp phát bộ nhớ cho một đối tượng ConstantValue.
  - Gán loại (type) là TP_INT.
  - Gán giá trị nguyên (intValue) bằng tham số đầu vào.
  - Trả về con trỏ đến đối tượng vừa được tạo.

16. void freeReferenceList(ObjectNode *objList)

```
264  void freeReferenceList(ObjectNode *objList) {
265    // TODO
266    // NOTE: tương tự như free object list
267    if (objList != NULL){
268        ObjectNode* temp;
269        while (objList->next != NULL){
270          temp = objList->next;
271          objList->next = temp->next;
272          free(temp);
273        }
274        free(objList);
275    }
276  }
```

- Ý nghĩa:
    - Tạo một hằng số nguyên (ConstantValue) và khởi tạo giá trị của nó.
- Cách thức hàm làm nhiệm vụ đó:
    - Cấp phát bộ nhớ cho một đối tượng ConstantValue.
    - Gán loại (type) là TP_INT.
    - Gán giá trị nguyên (intValue) bằng tham số đầu vào.
    - Trả về con trỏ đến đối tượng vừa được tạo.

17. Object* findObject(ObjectNode *objList, char *name)

```
292  Object* findObject(ObjectNode *objList, char *name) {
293    // TODO
294    ObjectNode* loop;
295    for (loop = objList; loop != NULL; loop = loop->next){
296      if (strcmp(loop->object->name, name) == 0){
297          return loop;
298      }
299    }
300    return NULL;
301  }
```

- Ý nghĩa:
    - Tạo một hằng số nguyên (ConstantValue) và khởi tạo giá trị của nó.
- Cách thức hàm làm nhiệm vụ đó:
    - Cấp phát bộ nhớ cho một đối tượng ConstantValue.
    - Gán loại (type) là TP_INT.
    - Gán giá trị nguyên (intValue) bằng tham số đầu vào.
    - Trả về con trỏ đến đối tượng vừa được tạo.