ΜΟΥΛΟΠΟΥΛΟΣ ΑΝΤΩΝΙΟΣ AM : 2104
ΧΡΟΝΑΡΑΚΗΣ ΑΡΙΣΤΕΙΔΗΣ AM : 2171

24/5/2015

ΠΛΥ 602 : "Μεταφραστες"

Project : Υλοποίηση μεταγλωττιστή για την γλωσσα Strange.

…..επεξήγηση ορισμένων σημείων του κώδικα με σκοπό την καλύτερη κατανόηση του τρόπου λειτουργίας του μεταγλωττιστή.....

- **#στον ενδιάμεσο κώδικα**

Δομες Δεδομένων

typedef struct namesOfSubprograms{ }Names: παιζει τον ρολο μιας λιστας οπου κρατάμε τα ονοματα των υποπρογραμμάτων του κυρίου προγράμματος.

typedef struct Quads { }Quad:ειναι μια δομή δεδομένων τα πεδία της οποίας χαρακτηρίζουν μια τετράδα του ενδιάμεσου κώδικα.

typedef struct ListOfQuads{ }List:παιζει τον ρολο μιας λιστας οπου κρατάμε τις τετράδες που δημιουργούνται στον ενδιάμεσο κώδικα.

Συναρτήσεις

void insertName(char *name){ }:συνάρτηση μέσω της οποίας περνάμε το όνομα ενος υποπρογράμματος στη λίστα οπου κρατάμε όλα τα ονόματα.

char *getNameOfSubProgram(){ }:συνάρτηση μέσω της οποίας λαμβάνουμε το τελευταίο στοιχείο που είχε προστεθεί στη λιστα με τα ονόματα δηλαδή το ονομα του τελευταίου υποπρογράμματος της λιστας.

void deleteLastName(){ }:συνάρτηση μέσω της οποίας αφου έχουμε λάβει το όνομα του τελευταίου υποπρογράμματος προβαίνουμε σε διαγραφή του απο τη λίστα ετσι ωστέ όταν ξανακαλέσουμε την getNameOfSubProgram() να πάρουμε το όνομα του προτελευταίου κ.ο.κ.

Quad *genQuad(char *op,char *x,char *y,char *z){ }: συνάρτηση μέσω της οποίας δημιουργείται μια τετράδα του ενδίάμεσου κώδικα.

void insert(Quad *aQuad){ }: συνάρτηση μέσω της οποίας αφου έχουμε δημιουργήσει μια τετράδα την εισάγουμε στη λιστα με όλες τις άλλες τετράδες.

char *newTemp(){ }: συνάρτηση η οποία μας επιστρέφει το όνομα της επόμενης προσωρινης μεταβλητής που μπορούμε να χρησιμοποιήσουμε.

List *emptyList(){ }: συνάρτηση μέσω της οποίας μας επιστρέφεται μια κένη λίστα τύπου List.

List *makeList(Quad *aQuad){ }: συνάρτηση μέσω της οποίας μας επιστρέφεται μια λίστα τύπου List η οποία περιέχει ενα μονο στοιχείο το Quad (την τετράδα) που πέρνει ως όρισμα.

List *merge(List *list1,List *list2){ }:συνάρτηση η οποία παίρνει ως όρισματα δυο λίστες και παει στο τελος της πρωτης και “κολλάει ” την δευτερη.

List *backPatch(List *list,int z){ }:συνάρτηση η οποία σε ολες τις τετράδες της λίστας που παίρνει ως όρισμα στο πεδίο z τους (των τετράδων) βάζει την τιμή του δευτέρου ορίσματος.

int nextquad(){}: συνάρτηση μέσω της οποίας μας επιστρέφεται ο αριθμος που πρέπει να αποδοθεί στο label της επόμενης τετράδας που θα δημιουργηθεί.

- **#στον πίνακα συμβόλων**

Δομες Δεδομένων

typedef struct RecordScope{ }Scope: παιζει τον ρολο μιας λιστας οπου κρατάμε το scope του κυρίου προγράμματος και των υποπρογραμμάτων του.

typedef struct RecordEntity{ }Entity: παιζει τον ρολο μιας λιστας οπου κρατάμε τις οντότητες (entities) ενος scope.

typedef struct RecordType{ }Type: ειναι μια δομή δεδομένων με δύο πεδία (τυπου Variable,τυπου Function).Αν πρόκειται για οντότητα που προσδιορίζει μια μεταβλητή τοτε το πεδιο Function θα εχει τιμη NULL.Αντίστοιχα αν πρόκειται για οντότητα που προσδιορίζει ενα υποπρόγραμμα τότε το πεδίο Variable θα ειναι NULL.

typedef struct RecordVariable{ }Variable: ειναι μια δομή δεδομένων που προσδιορίζει τα επιπλεον πεδία μιας οντότητας στην περίπτωση που ο τυπος της ειναι Variable δηλαδή πρόκειται για οντότητα μεταβλητή.

typedef struct RecordFunction{ }Function: ειναι μια δομή δεδομένων που προσδιορίζει τα επιπλεον πεδία μιας οντότητας στην περίπτωση που ο τυπος της ειναι Function δηλαδή πρόκειται για οντότητα υποπρόγραμμα.

typedef struct RecordArgument{ }Argument: παιζει τον ρολο μιας λιστας οπου κρατάμε ολες τις παραμέτρους ενός υποπρογράμματος.

Συναρτήσεις

Scope *createScope(){ }: συνάρτηση η οποία προσθέτει ένα νέο scope στη λιστα με τα scopes το οποίο μάλιστα μπορεί και να το επιστρέφει.

void deleteScope(){ }: συνάρτηση μέσω της οποίας αφου έχουμε τελειώσει με την αναλυση ενος scope (υποπρογραμματος) προβαίνουμε σε διαγραφή του απο τη λίστα με τα scopes.

void insertScope(Entity *anEntity){ }: συνάρτηση μέσω της οποίας κάνουμε εισαγωγή την λιστα απο οντότητες που παιρνει ως όρισμα στο χρονικα τελευταίο scope που προστέθηκε στη λίστα απο τα scopes.

Entity *getEntity(){ }: συνάρτηση μέσω της οποίας μας επιστρέφεται η λιστα των οντοτήτων του χρονικα τελευταίου scope που προστέθηκε στη λίστα απο τα scopes.

Entity *insertEntity(Entity *anEntity,int type){ }: συνάρτηση μέσω της οποίας στο τελος της λιστας που παιρνει ως όρισμα εισάγει μια νέα οντότητα και της δινει τιμές στα πεδία της ανάλογα με το τι τυπος οντότητας ειναι(μεταβλητη/υποπρογραμμα) το οποιο καθοριζεται απο το δευτερο ορισμα της συναρτησης(type).Η λιστα των οντοτήτων επιστρεφεται απο την συναρτηση.

Entity *insertTempVar(Entity *anEntity,char *var){ }: συνάρτηση μέσω της οποίας στο τελος της λιστας που παιρνει ως όρισμα για τις προσωρινές μεταβήτες που χρησιμοποιούμε εισάγει μια νέα οντότητα που ο τύπος της ειναι μεταβλητή και της δίνει και τιμές.Η λιστα των οντοτήτων επιστρεφεται απο την συναρτηση.

Argument *insertArgument(Argument *argList){ }: συνάρτηση μέσω της οποίας στο τελος της λιστας των παραμέτρων που παιρνει ως όρισμα εισάγει ενα νέο Argument (παράμετρο) στο οποίο δίνει και τιμές ανάλογα με τον τρόπο περάσματος της παραμέτρου αυτής.Η λιστα των παραμετρων επιστρεφεται απο την συναρτηση.

Entity *insertArgumentListToEntity(Entity *entity,Argument *argList){ }: συνάρτηση μέσω της οποίας στο τελευταίο στοιχείο της λιστας του πρωτου ορίσματος που ειναι οντότητα τυπου Function πάει και “βάζει” στο κατάλληλο πεδίο της την λίστα με τις παραμέτρους του δευτέρου ορίσματος.Η λιστα των οντοτήτων επιστρεφεται απο την συναρτηση.

void calculateFrame(){ }: συνάρτηση μέσω της οποίας γίνεται ο υπολογισμός του framelength των υποπρογραμμάτων.

int getStartQuad(){ }: συνάρτηση που μας επιστρέφει το startQuad (εναρκτήρια τετράδα) ενός υποπρογράμματος.

Entity *search(char *varName,int *aChoice,int *nestingLevel){ }: συνάρτηση μέσω της οποίας γίνεται αναζήτηση με βαση το varName σε ολα τα scopes απο το τελευταίο χρονικά που εισήχθει εως το scope του κύριου προγράμματος για μια οντότητα με τύπο μεταβλητή της οποία το όνομα ταυτίζεται με το varName.Αν βρεθεί τότε επιστρέφει την οντότητα αυτή και αλλάζει και με αναφορά την τιμή της παραμέτρου aChoice αναλόγως με το τι ισχύει για την μεταβλητή.Πιο συγκεκριμένα

choice=0 :αν πρόκειται για global μεταβλητή.

choice=1 :αν πρόκειται για τοπική μεταβλητή ή τυπική παράμετρο που περνά με τιμή και βαθος φωλιάσματος ίσο με το τρέχον ή προσχωρινή μεταβλητή.

choice=2 :αν πρόκειται για τυπική παράμετρο που περνά με αναφορά και βάθος φωλιάσματος ίσο με το τρέχον.

choice=3: αν πρόκειται για τοπική μεταβλητή ή τυπική παράμετρο που περνά με τιμή και βαθος φωλιάσματος μικρότερο απο το τρέχον.

choice=4:αν πρόκειται για τυπική παράμετρο που περνά με αναφορά και βάθος φωλιάσματος μικρότερο απο το τρέχον.

Entity *searchFunction(char *funName,int *aChoice){ }:συνάρτηση μέσω της οποίας γίνεται αναζήτηση με βαση το funName σε ολα τα scopes απο το τελευταίο χρονικά που εισήχθει εως το scope του κύριου προγράμματος για μια οντότητα με τύπο υποπρόγραμμα (Function) της οποία το όνομα ταυτίζεται με το funName.Αν βρεθεί τότε επιστρέφει την οντότητα αυτή και αλλάζει και με αναφορά την τιμή της παραμέτρου aChoice αναλόγως με το τι ισχύει για την μεταβλητή.Πιο συγκεκριμένα

choice=0 :αν πρόκειται για global μεταβλητή.
choice=1 :αν πρόκειται για τοπική μεταβλητή ή τυπική παράμετρο που περνά με τιμή και βαθος φωλιάσματος ίσο με το τρέχον ή προσχωρινή μεταβλητή.
choice=2 :αν πρόκειται για τυπική παράμετρο που περνά με αναφορά και βάθος φωλιάσματος ίσο με το τρέχον.
choice=3: αν πρόκειται για τοπική μεταβλητή ή τυπική παράμετρο που περνά με τιμή και βαθος φωλιάσματος μικρότερο απο το τρέχον.
choice=4:αν πρόκειται για τυπική παράμετρο που περνά με αναφορά και βάθος φωλιάσματος μικρότερο απο το τρέχον.

- **#στον τελικο κώδικα**

Συναρτήσεις

void checkQuad(int start){ }: συνάρτηση μέσω της οποίας ελέγχουμε ολές τις τετραδες ενδιάμεσου κώδικα ενός υποπρογράμματος και αναλόγως με την φύση κάθε τετράδας προχωρούμε στην συγγραφή τελικού κώδικα για την τετράδα αυτη.

void checkMainQuad(){ }:συνάρτηση μέσω της οποίας ελέγχουμε ολές τις τετραδες ενδιάμεσου κώδικα του κυρίως προγράμματος και αναλόγως με την φύση κάθε τετράδας προχωρούμε στην συγγραφή τελικού κώδικα για την τετράδα αυτη.

void deleteQuads(int aLabel){ }:συνάρτηση μέσω της οποίας αφού έχουμε ελεγξει όλες τις τετράδες ενός υποπρογράμματος και εχουμε δημιουργήσει τελικό κώδικα για το υποπρογραμμα αυτό προβαίνουμε σε διαγράφη των τετράδων ενδιάμεσου κώδικα για το υποπρογραμμα αυτό απο την γενική λίστα των τετράδων.

...καποια σχόλια για την ομαλή εκτέλεση του κώδικα...

[1]για την εκτέλεση αριθμητικών παραστάσεων θα πρέπει μεταξύ του τελούμενου και του τελεστή να υπάρχει κενό (whitespace/tab/return) για να την αντιλαμβάνεται ως αριθμητική παράσταση.Για παράδειγμα θα πρέπει η αριθμητική πράξη να ειναι της μορφής 3 * 2και οχι 3*2.

[2]για την δημιουργία 4αδας οταν μια παράμετρος μεταδίδεται με αντιγραφή χρησιμοποιήσαμε στο ζητούμενο πεδίο της τετράδας το "CPY".Για παράδειγμα genQuad("par",term,"CPY","").

[3]για τις υλοποιήσεις των incase & forcase τα when που χρησιμοποιούνται πρέπει να είναι εντός παρενθέσεων για σωστή υλοποίηση .Για παράδειγμα forcase{ (when([a=0]){c:=0}) } και όχι forcase{when([a=0]){c:=0}}.

[4]

```
program first
{

    declare a,b,m,n,o,p,c,k,l,d,v,h,y enddeclare



    procedure p1(inout c2,in c3){

        procedure p2(inout c4){
            print(c4)
        }

        call p2(inout c2);
        print(c2);
        print(c3)

    }



    a := 15;
    b := 20;

    call p1(inout a,in b);


    print(a)


}
```

Στο παραπάνω παράδειγμα το c3 ενω έπρεπε να εμφανίσει 20 εμφανιζει 67. Ενώ στο παρακάτω παράδειγμα που είναι το ίδιο με τη διαφορά ότι η p2 παίρνει δυο ορίσματα τα αποτελέσματα είναι τα σωστά.

```
program first
{

    declare a,b,m,n,o,p,c,k,l,d,v,h,y enddeclare
```

```
procedure p1(inout c2,in c3){

        procedure p2(inout c4,inout c5){
                print(c4)
        }

        call p2(inout c2, inout c3);
        print(c2);
        print(c3)

}



a := 15;
b := 20;

call p1(inout a,in b);


print(a)


}
```

[5]στις υλοποιήσεις των incase & forcase όταν ο αριθμός των when που χρησιμοποιούνται είναι διαφορετικός των τριών έχουμε σωστή υλοποίηση.Στην περίπτωση που τα when ειναι τρία το πρόγραμμα μπαίνει σε ατερμών βρόχο.