Assignment 2: "A String Module"

Σκοπός

Ο σκοπός αυτής της άσκησης είναι να σας βοηθήσει να κάνετε επανάληψη/μάθετε (1) τη χρήση arrays και pointers στη γλώσσα προγραμματισμού C, (2) πως να δημιουργείτε modules χωρίς state στη C, και (3) τη χρήσης των εργαλείων προγραμματισμού GNU/UNIX, ειδικά gcc, shell, και editing (emacs).

Background

Όπως γνωρίζετε, το περιβάλλον της γλώσσας προγραμματισμού C περιέχει μια standard βιβλιοθήκη (standard C library). Οι συναρτήσεις που παρέχει η standard C library δηλώνονται σε header files. Ένα από αυτά τα header files είναι το string.h, που περιέχει τους ορισμούς των συναρτήσεων για την εκτέλεση λειτουργιών σε strings (C string functions). Οι σελίδες των manuals του UNIX (man pages) περιγράφουν τις συναρτήσεις αυτές. Πληκτρολογήστε "man function_name" για να δείτε πληροφορίες για κάθε συνάρτηση. Οι συναρτήσεις string της C χρησιμοποιούνται συχνά σε πολλά συστήματα, όπως editors, assemblers, compilers, operating systems, κ.τ.λ.

Η άσκηση

Η άσκηση αυτή σας ζητάει να χρησιμοποιήσετε την γλώσσα C για να δημιουργήσετε το module "MyString", το οποίο θα περιέχει υλοποιήσεις των πιο συχνά χρησιμοποιούμενων συναρτήσεων string. Ειδικότερα, το module MyString θα περιέχει τις παρακάτω συναρτήσεις, που θα συμπεριφέρονται όπως οι αντίστοιχες συναρτήσεις στη C:

Συνάρτηση MyString	Αντίστοιχη συνάρτηση στη standard C library
ms_length	strlen
ms_copy	strcpy
ms_ncopy	strncpy
ms_concat	strcat
ms_nconcat	strncat
ms_compare	strcmp
ms_ncompare	strncmp
ms_search	strstr

Λεπτομέρειες

- Σχεδιάστε το πρόγραμμά σας σύμφωνα με ένα καλά ορισμένο «συμβόλαιο» για κάθε συνάρτηση. Πριν από κάθε συνάρτηση γράψτε ένα σχόλιο το οποίο περιγράφει τι κάνει η συνάρτηση (όχι πως το κάνει) και τι λάθη ελέγχει η συνάρτηση κατά την εκτέλεσή της (checked runtime errors). Κάθε συνάρτηση πρέπει να χρησιμοποιεί το macro assert (ορισμένο στο header file assert.h) ώστε να ελέγχει τα λάθη αυτά. Ειδικότερα, οι συναρτήσεις σας πρέπει να ελέγχουν ότι κάθε array/pointer formal parameter της συνάρτησης δεν είναι NULL.

  Δεν θα χρειαστεί να προσθέσετε άλλες κλήσεις του macro assert στον κώδικά σας, ωστόσο σκεφτείτε και απαντήστε στο readme αρχείο της άσκησης αν είναι δυνατό να προσθέσετε ελέγχους για:

  1. Είναι δυνατό οι συναρτήσεις ms_copy, ms_ncopy, ms_concat, ms_nconcat να καλέσουν την assert και να ελέγξουν αν η μνήμη προορισμού όπου γράφεται το αποτέλεσμα είναι αρκετά μεγάλη; Εξηγήστε.
  2. Είναι δυνατό οι συναρτήσεις ms_ncopy, ms_nconcat, ms_ncompare να καλέσουν την assert και να ελέγξουν αν η παράμετρος length έχει αρνητική τιμή; Εξηγήστε.

- Ορίστε το "interface" του module MyString σε ένα header file mystring.h. Μπορείτε στους ορισμούς αυτούς να χρησιμοποιήσετε είτε συμβολισμό arrays είτε συμβολισμό pointers (θυμηθείτε ότι αυτά είναι ισοδύναμα όταν πρόκειται για ορισμό παραμέτρων συναρτήσεων). Γράψτε **δύο** υλοποιήσεις του **ενός** interface που ορίσατε για το module MyString.

  Κάντε την πρώτη υλοποίηση σε ένα αρχείο mystring_ars.c. Η υλοποίηση αυτή πρέπει να περιέχει ορισμούς των συναρτήσεων του module χρησιμοποιώντας συμβολισμό arrays και όχι pointers. Π.χ. στην υλοποίηση αυτή μπορεί να ορίσετε τη συνάρτηση ms_length ως:

```
size_t ms_length(const char pcStr[])
{
  size_t uiLength= 0U;
  assert(pcStr != NULL);
  while (pcStr[uiLength] != '\0')
    uiLength++;
```

```
 return uiLength;
}
```

(Ο τύπος size_t ορίζεται στο stddef.h. Είναι ένας unsigned ακέραιος τύπος που εξαρτάται από το κάθε σύστημα και είναι αρκετά μεγάλος να αναπαραστήσει το μήκος οποιουδήποτε string στο σύστημα. Συνήθως ορίζεται ως "unsigned int" ή ως "unsigned long". Διάφορες συναρτήσεις string της C χρησιμοποιούν τύπο αυτό οπότε και οι δικές σας θα πρέπει να κάνουν το ίδιο πράγμα.)

Η δεύτερη υλοποίηση θα βρίσκεται στο αρχείο mystring_ptrs.c. Θα πρέπει να περιέχει ορισμούς των συναρτήσεων του module MyString χρησιμοποιώντας συμβολισμό pointers (και όχι arrays). Π.χ. σε αυτή την υλοποίηση μπορεί να ορίσετε τη συνάρτηση ms_length ως:

```
size_t ms_length(const char *pcStr)
{
 size_t uiLength = 0U;
 assert(pcStr != NULL);
 while (*(pcStr + uiLength) != '\0')
  uiLength++;
 return uiLength;
}
```

Σε αυτή την υλοποίηση προσπαθήστε να υλοποιήσετε τις συναρτήσεις με πιο αποτελεσματικό τρόπο (από άποψη χρόνου εκτέλεσης), από την απλή μετάφραση του "a[i]" σε "*(a+i)". Π.χ.

```
size_t ms_length(const char *pcStr)
{
 const char *pcStrEnd = pcStr;
 assert(pcStr != NULL);
 while (*pcStrEnd != '\0')
  pcStrEnd++;
 return pcStrEnd - pcStr;
}
```

- Οι συναρτήσεις σας και στις δύο υλοποιήσεις του module MyString, δεν πρέπει να καλούν καμία υπάρχουσα συνάρτηση από τις string functions της C, αλλά να προσποιούνται ότι αυτές οι συναρτήσεις δεν υπάρχουν. Ωστόσο οι συναρτήσεις σας μπορούν να καλούν η μια την άλλη και μπορείτε να ορίσετε και επιπλέον συναρτήσεις.

  Προσέξτε τις οριακές περιπτώσεις στον κώδικα σας, π.χ. ότι οι συναρτήσεις σας δουλεύουν σωστά με άδεια string ως παραμέτρους. Π.χ. η κλήση ms_length("") πρέπει να επιστρέφει 0, η κλήση ms_length(NULL) να επιστρέφει error, κ.τ.λ. Δώστε προσοχή σε type mismatches, όπως π.χ. κατά την χρήση του keyword const. Προσέξτε ότι μια μεταβλητή τύπου size_t μπορεί να αναπαραστήσει μεγαλύτερους ακεραίους από μια μεταβλητή τύπου int.

Συστάσεις

- Πολλές φορές οι προγραμματιστές σε μια γλώσσα προγραμματισμού χρησιμοποιούν ειδικές «ιδιωματικές» εκφράσεις που κάνουν τον κώδικα πιο ευανάγνωστο (ειδικά σε όποιον είναι συνηθισμένος σε αυτά τα ιδιώματα). Π.χ. στη C η αναπαράσταση του FALSE είναι η ίδια με την αναπαράσταση του NULL χαρακτήρα που τερματίζει τα string. Αν θέλετε, μπορείτε να χρησιμοποιείτε τέτοιου είδους ιδιώματα στον κώδικα σας. Π.χ. μπορείτε να γράψετε τη συνάρτηση ms_length ως:

```
size_t ms_length(const char pcStr[])
{
 size_t uiLength = 0U;
 assert(pcStr); /* Works because NULL and FALSE are identical. */
 while (pcStr[uiLength]) /* Works because end-of-string and */
  uiLength++;          /* FALSE are identical. */
 return uiLength;
}
```

ή ως:

```
size_t ms_length(const char *pcStr)
{
 const char *pcStrEnd = pcStr;
 assert(pcStr); /* Works because NULL and FALSE are identical. */
 while (*pcStrEnd) /* Works because end-of-string and FALSE are identical. */
```

```
  pcStrEnd++;
 return pcStrEnd - pcStr;
}
```

Ωστόσο, αποφύγετε την χρήση ιδιωματικών εκφράσεων που ίσως κάνουν το πρόγραμμά σας πιο δύσκολο στην κατανόηση του (από σας ή από τρίτους).

- Δώστε προσοχή στα σχόλια του προγράμματος σας.
    - Ξεκινήστε κάθε αρχείο που δημιουργείτε με ένα σχόλιο που περιλαμβάνει το όνομά σας, τον αριθμό της άσκησης, και το όνομα του αρχείου.
    - Ξεκινήστε κάθε συνάρτηση του προγράμματος σας με ένα σχόλιο που περιγράφει τι κάνει ο υπολογιστής όταν εκτελεί τη συνάρτηση. Το σχόλιο πρέπει να χρησιμοποιεί στην περιγραφή αυτή τα ονόματα των παραμέτρων της συνάρτησης, όπου αυτό είναι δυνατό. Το σχόλιο πρέπει να αναφέρει ρητά αν και τι διαβάζει η συνάρτηση από το stdin (ή οποιοδήποτε άλλο stream) και αν και τι γράφει στο stdout (ή σε οποιοδήποτε άλλο stream). Επίσης το σχόλιο πρέπει να αναφέρει τι runtime errors ελέγχει η συνάρτηση (checked runtime errors). Επίσης, είναι επιθυμητό το σχόλιο αυτό να υπάρχει και στο .h file που **δηλώνει** τη συνάρτηση, ώστε να το βλέπουν αυτοί που χρησιμοποιούν τη συνάρτηση, και στο .c file που **ορίζει/υλοποιεί** τη συνάρτηση, ώστε να το βλέπουν αυτοί που γράφουν/συντηρούν τον κώδικα. Π.χ.

      Στο αρχείο mystring.h:

```
...

/* Return the length of string pcStr.
   It is a checked runtime error for pcStr to be NULL. */
size_t ms_length(const char pcStr[]);

...
```

      Στο αρχείο mystring_ptrs.c:

```
...
/* Return the length of string pcStr.
   It is a checked runtime error for pcStr to be NULL. */
size_t ms_length(const char pcStr[])
{
  const char *pcStrEnd = pcStr;
  assert(pcStr != NULL);
  while (*pcStrEnd != '\0')
    pcStrEnd++;
  return pcStrEnd - pcStr;
}
...
```

- Η χρήση του keyword **const** είναι γενικά κάπως λεπτή. Δεδομένου ότι οι συναρτήσεις σας πρέπει να δηλωθούν όπως οι αντίστοιχες συναρτήσεις στη C, προσπαθήστε κατά την υλοποίηση των συναρτήσεων να δηλώσετε τις επιπλέον μεταβλητές σας κατά τέτοιο τρόπο ώστε να μην χρειάζεται να κάνετε μετατροπές τύπων (type casting) και να μην έχετε warnings από τον compiler είτε κατά την μετάφραση του προγράμματος σας, είτε κατά την μετάφραση προγραμμάτων που χρησιμοποιούν τις συναρτήσεις σας. Αυτό πρέπει να είναι δυνατό σε όλες τις περιπτώσεις εκτός ίσως από τη συνάρτηση ms_search.

Logistics

- Γράψτε το πρόγραμμα σας στα μηχανήματα, αρχιτεκτονικής x86, του τμήματος με λειτουργικό σύστημα GNU/Linux (portokali, milo, rodakino, etc.) χρησιμοποιώντας τα εργαλεία gcc, emacs/vim/nano, gdb.
- Δημιουργήστε με τον αγαπημένο σας κειμενογράφο τα αρχεία της άσκησης (mystring.h, mystring_ars.c, mystring_ptrs.c). Περιορίστε το μέγεθος των γραμμών στο αρχείο σας σε 78 ή 80 χαρακτήρες. Αυτό σας επιτρέπει να τυπώνετε σε δύο στήλες σε χαρτί και να έχετε ταυτόχρονα ανοιχτά παράθυρα για editing και compilation και execution.
- Χρησιμοποιήστε τον gcc με τις command line παραμέτρους "-Wall, -ansi, -pedantic" για να κάνετε preprocess, compile, assemble, και link το πρόγραμμά σας:
- Δημιουργήστε ένα "readme" text file που περιέχει:
    1. Το όνομα σας.
    2. Τις απαντήσεις στις δύο ερωτήσεις παραπάνω.
    3. Πράγματα που χειρίζεστε με διαφορετικό τρόπο από ότι ορίζει η άσκηση.

4. Μια περιγραφή της βοήθειας που είχατε από άλλους στη δημιουργία του προγράμματος σας, και σε συμφωνία με το "Policies" section του web page του μαθήματος.
5. (Προαιρετικά) Μία ένδειξη του πόσο χρόνο αφιερώσατε για την άσκηση.
6. (Προαιρετικά) Οτιδήποτε άλλο θέλετε να αναφέρετε.

- Σχόλια που περιγράφουν τον κώδικά σας δεν πρέπει να υπάρχουν στο readme file. Πρέπει να τα ενσωματώσετε στο κατάλληλο σημείο του προγράμματος σας.

Υποβολή

Υποβάλετε την άσκηση σας όπως αναφέρεται στο section Policies του web page του μαθήματος.

Βαθμολογία

Η βαθμολογία θα βασιστεί και στην ορθότητα αλλά και στο σχεδιασμό, όπως αναφέρεται στη σελίδα Policies του μαθήματος. Η κατανόηση της άσκησης αλλά και η αναγνωσιμότητα ενός προγράμματος είναι σημαντικό μέρος του σχεδιασμού.

Last Modified: 22-06-2015 14:36