Η Βιβλιοθήκη i2p

1. Βιβλιοθήκη τρίτου κατασκευαστή (Third-Party library)

Third-Party library είναι μια βιβλιοθήκη που έχει αναπτυχθεί από άλλον προγραμματιστή. Η σημασία αξιοποίησης βιβλιοθήκης τρίτου κατασκευαστή είναι εμφανής από το παρακάτω απόσπασμα.

"In computer programming, a **third-party software component** is a reusable software component developed to be either freely distributed or sold by an entity other than the original vendor of the development platform. The third-party software component market thrives because many programmers believe that **component-oriented development** improves the efficiency and the quality of developing custom applications." Wikipedia

2. Αντικείμενο, Στόχος και Εκδόσεις της βιβλιοθήκη i2p

Η βιβλιοθήκη i2p αναπτύχθηκε για δύο λόγους:

1. για να σας δώσει τη δυνατότητα να εξοικειωθείτε με τη διαδικασία αξιοποίησης βιβλιοθήκης τρίτου κατασκευαστή, και,
2. να σας διευκολύνει στην ανάπτυξη των πρώτων εκδόσεων ορισμένων προγραμμάτων.

Η βιβλιοθήκη σας δίνεται σε τρεις (3) εκδόσεις.

2.1. 1η έκδοση της i2p

Η **πρώτη** έκδοση από αυτές αναπτύχθηκε για να χρησιμοποιηθεί στις πρώτες εκδόσεις της άσκησης Fractions. Ένας από τους βασικούς στόχους αυτής της άσκησης είναι να εξοικειωθείτε με τη διαδικασία αξιοποίησης βιβλιοθήκης τρίτου κατασκευαστή.
Η πρώτη έκδοση σας δίνεται με την μορφή 2 αρχείων: Το αρχείο libi2p.dll που περιέχεται στο libi2p.zip και το αντίστοιχο αρχείο επικεφαλίδας (header file) (αρχείο i2p.h).

2.2. 2η έκδοση της i2p

Η **δεύτερη** έκδοση αναπτύχθηκε για να χρησιμοποιηθεί στην εκδοχή της άσκησης Fractions που αξιοποιεί την κατασκευή της δομής (struct). Η έκδοση αυτή σας δίνεται με την μορφή ενός συμπιεσμένου αρχείου (libi2p.zip) που περιέχει το .dll και το .h αρχείο.

2.3. 3η έκδοση της i2p

Η **τρίτη** έκδοση ενσωματώνει στην δεύτερη έκδοση και συναρτήσεις ταξινόμησης πίνακα ακεραίων και αλφαριθμητικών για χρήση στις αντίστοιχες ασκήσεις μεταξύ των οποία η Ταξινόμηση Λέξεων και ταξινόμηση πίνακα ακεραίων. Η έκδοση αυτή σας δίνεται με την μορφή ενός συμπιεσμένου αρχείου (libi2pV3.zip) που περιέχει το .dll και το .h αρχείο.

3. Αξιοποίηση βιβλιοθήκης

Για να αξιοποιήσετε μια βιβλιοθήκη θα πρέπει να κάνετε τις παρακάτω ενέργειες:

3.1. E1-Κατεβάστε την βιβλιοθήκη

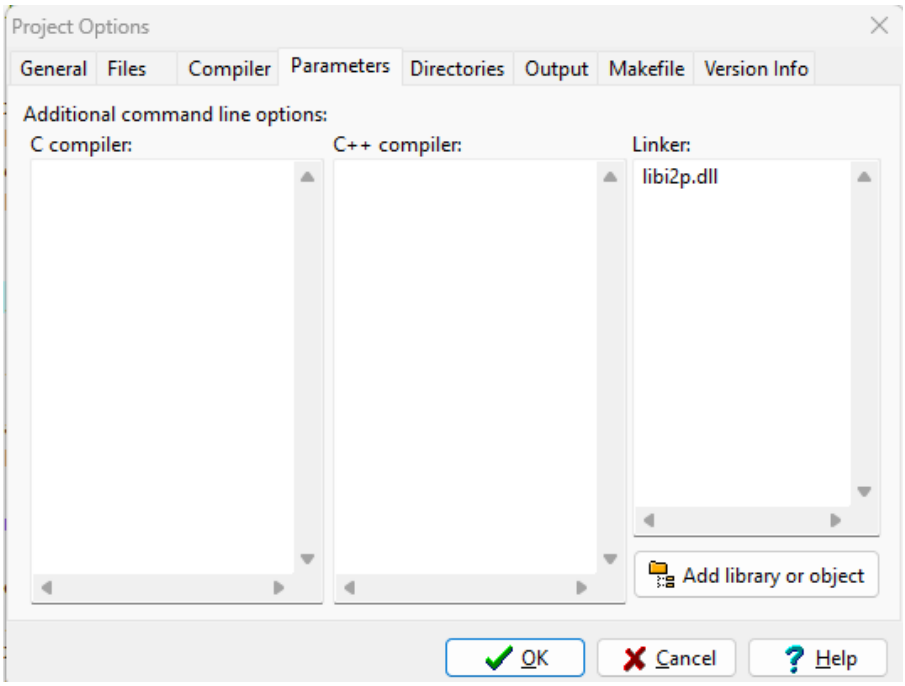
Κατεβάστε και τοποθετήστε την βιβλιοθήκη (αρχείο .h και αρχείο .dll) στο ευρετήριο του project σας.

3.2. E2-Συμπεριλάβετε το αρχείο επικεφαλίδας (#include)

Συμπεριλάβετε με την εντολή προεπεξεργαστή #include στο αρχείο πηγαίου κώδικα σας το αρχείο επικεφαλίδας, π.χ. #include “i2p.h”.
Τοποθετούμε το header file σε “ ” για να ψάξει ο προεπεξεργαστής (pre-processor) το αρχείο στο φάκελο (ευρετήριο) του προγράμματος μας.
Σε < > περικλείονται τα αρχεία επικεφαλίδας της βασικής βιβλιοθήκης για να δηλώσουν στον επεξεργαστή πως αυτά θα τα αναζητήσει στον φάκελο του compiler που χρησιμοποιείτε.

3.3. E3-Ενημερώστε το IDE σας για το αρχείο .dll

Για παράδειγμα για να ενημερώσετε το DevC++ επιλέξτε
Project->Projects Options>Parameters και
στην συνέχεια `Add library or Object` (Σχήμα 1). Επιλέξτε από το ευρετήριο του project το αρχείο της βιβλιοθήκης. Θα το δείτε στο πλαίσιο κάτω από τον Linker καθώς το αρχείο αυτό θα το αξιοποιήσει, μετά την μεταγλώττιση, ο Linker. Επιλέξτε OK.



Σχήμα 1. Ενημέρωση IDE για το αρχείο lib12p.dll

3.4. E4-Καλέστε συνάρτηση της βιβλιοθήκης

Καλέστε μια συνάρτηση της βιβλιοθήκης πάντα σύμφωνα με το function prototype της που θα βρείτε στο αρχείο επικεφαλίδας της βιβλιοθήκης. Στην επόμενη ενότητα δίνονται οι συναρτήσεις που περιέχει η βιβλιοθήκη και στην μεθεπόμενη οδηγίες και παραδείγματα κλήσης των συναρτήσεων.

4. Οι συναρτήσεις της βιβλιοθήκης i2p

4.1. Συναρτήσεις 1ης έκδοσης της βιβλιοθήκης (`readExpression V1`)

Η 1η έκδοση περιέχει μία μόνο συνάρτηση την `readExpression()`.
Το function prototype της συνάρτησης, το οποίο βρίσκεται στο αρχείο επικεφαλίδας (header file) i2p.h είναι το παρακάτω:

```
void readExpression(char *operatorPtr, int *op1nPtr,int *op1dPtr,
                    int *op2nPtr,int *op2dPtr);
```

όπου

`operatorPtr` : operator Pointer (Δείκτης στην μεταβλητή operator)

`op1nPtr` : operand1 numerator Pointer (Δείκτης στην μεταβλητή operand1 numerator)

`op1dPtr` : operand1 denominator Pointer (Δείκτης στην μεταβλητή operand1 denominator)

`op2nPtr` : operand2 numerator Pointer

`op2dPtr` : operand2 denominator Pointer

Η συνάρτηση διαβάζει τα στοιχεία που αποτελούν μια έκφραση προθεματικού τελεστή (prefix notation) (παράγραφος 4.2.1) με κλάσματα, όπως για παράδειγμα η + 2/13 5/13. Διαβάζει δηλαδή τον τελεστή και τους αριθμητές και παρονομαστές των κλασμάτων. Επειδή δεν μπορεί με τον μηχανισμό της επιστρεφόμενης τιμής να επιστρέψει όλα αυτά στην συνάρτηση που την κάλεσε, η συνάρτηση δέχεται ως ορίσματα δείκτες σε μεταβλητές όπου θα βάλει τις τιμές που θα διαβάσει από την κύρια είσοδο. Τον περιορισμό αυτό που βάζει ο μηχανισμός της επιστρεφόμενης τιμής θα τον παρακάμψουμε στην επόμενη έκδοση της συνάρτησης η οποία θα αξιοποιεί τις δομές (struct).

[Δράση 1] Αξιοποιήστε την έκδοση αυτή της βιβλιοθήκης (δηλαδή την `readExpression`()) για να αναπτύξετε ένα πρόγραμμα που θα δίνει τη δυνατότητα στη μηχανή να υπολογίζει τιμές εκφράσεων με κλάσματα (Άσκηση Fractions).

4.2. Συναρτήσεις 2ης έκδοσης της βιβλιοθήκης (`readExpression V2`)

Η 2η έκδοση περιέχει και αυτή μία μόνο συνάρτηση την `readExpression()`.
Το function prototype της συνάρτησης, το οποίο βρίσκεται στο αρχείο επικεφαλίδας (header file) i2p.h είναι το παρακάτω:

```
Expression readExpression(void);
```

Παρατηρούμε πως η έκδοση αυτή της συνάρτησης δεν δέχεται κανένα όρισμα αλλά αξιοποιεί την κατασκευή της δομής για να επιστρέψει στην συνάρτηση που την κάλεσε τα στοιχεία της έκφρασης τα οποία διαβάζει από την κύρια είσοδο (stdin).

Το αρχείο επικεφαλίδας i2p.h εκτός από το πρωτότυπο της συνάρτησης περιέχει και τον ορισμό της δομής `Expression` ο οποίος δίνεται στο Σχήμα 2.

i2p.h

```
1   #include <search.h>
2
3   typedef struct fraction{
4       int ar;
5       int par;
6   }Fraction;
7
8   typedef struct expression{
9       char operator;
10      Fraction op1;
11      Fraction op2;
12  }Expression;
13
14  Expression readExpression(void); //
```

Σχήμα 2. Το αρχείο επικεφαλίδας i2p.h της 2ης έκδοσης της βιβλιοθήκης i2p.

[Δράση 2] Αξιοποιήστε την έκδοση αυτή της βιβλιοθήκης για να δώσετε μια νέα έκδοση στο πρόγραμμα σας Fractions το οποίο να αξιοποιεί εκτός από την `readExpression` () και τις δομές `Fraction` `Expression` που αυτή ορίζει.

4.3. Συναρτήσεις 3ης έκδοσης της βιβλιοθήκης

Η 3η έκδοση περιέχει ότι και η 2η έκδοση και επιπλέον περιέχει συναρτήσεις για ταξινόμηση πίνακα αλφαριθμητικών και πίνακα ακεραίων.

Τα function prototypes των συναρτήσεων ταξινόμησης πίνακα αλφαριθμητικών, τα οποία βρίσκονται στο αρχείο επικεφαλίδας (header file) i2p.h είναι τα παρακάτω:

```
void sortInc4String(char *base, int numOfElements, int strWidth);
void sortDec4String(char *base, int numOfElements, int strWidth);
```

Η συνάρτηση `sortInc4String` ταξινομεί ένα πίνακα αλφαριθμητικών με αύξουσα σειρά ενώ η `sortDec4String` με φθίνουσα. Και οι δύο συναρτήσεις δέχονται ως ορίσματα τα:

`base`: δείκτης στο πρώτο αλφαριθμητικό του πίνακα.

`numOfElements`: ο αριθμός των στοιχείων (αλφαριθμητικών) του πίνακα.

`strWidth`:Το μέγεθος σε bytes του στοιχείου του πίνακα, δηλαδή του αλφαριθμητικού.

Τα function prototypes των συναρτήσεων ταξινόμησης πίνακα ακεραίων, τα οποία βρίσκονται στο αρχείο επικεφαλίδας (header file) i2p.h είναι τα παρακάτω:

```
void sortInc4Int(int *base, int numOfElements, int width);
void sortDec4Int(int *base, int numOfElements, int width);
```

Η συνάρτηση `sortInc4Int` ταξινομεί ένα πίνακα ακεραίων με αύξουσα σειρά ενώ η `sortDec4Int` με φθίνουσα. Και οι δύο συναρτήσεις δέχονται ως ορίσματα τα:

`base`: δείκτης στο πρώτο στοιχείο (ακέραιο) του πίνακα.

`numOfElements`: ο αριθμός των στοιχείων (ακεραίων) του πίνακα.

`strWidth`: το μέγεθος σε bytes του στοιχείου του πίνακα, δηλαδή του ακεραίου.

Το αρχείο επικεφαλίδας i2p.h της 3ης έκδοσης της i2p δίνεται στο Σχήμα 3.

i2p.h

```
1   #include <search.h>
2
3   typedef struct fraction{
4       int ar;
5       int par;
6   }Fraction;
7
8   typedef struct expression{
9       char operator;
10      Fraction op1;
11      Fraction op2;
12  }Expression;
13
14  Expression readExpression(void); // reads an expression of the fo
15
16  void sortInc4String(char *base,int numOfElements, int strWidth);
17  void sortDec4String(char *base,int numOfElements, int strWidth);
18
19  void sortInc4Int(int *base,int numOfElements, int width);
20  void sortDec4Int(int *base,int numOfElements, int width);
```

Σχήμα 3. Το αρχείο επικεφαλίδας i2p.h της 3ης έκδοσης της βιβλιοθήκης i2p.

[Δράση 3] Αξιοποιήστε την έκδοση αυτή της βιβλιοθήκης για να ταξινομήσετε σε αύξουσα και φθίνουσα σειρά ένα πίνακα ακεραίων και ένα πίνακα αλφαριθμητικών.

5. Παραδείγματα κλήσης συνάρτησης

Η βιβλιοθήκη i2p περιέχει την συνάρτηση readExpression() η οποία και στις 2 εκδόσεις της :

a) ζητάει από τον χρήση να εισάγει από την βασική είσοδο μια έκφραση όπως αυτή ορίζεται από την άσκηση Fractions.
b) Διαβάζει από την βασική είσοδο τα στοιχεία της έκφρασης που εισάγει ο χρήστης.

Στα στοιχεία αυτά θα πρέπει με κάποιο τρόπο να αποκτήσει πρόσβαση η συνάρτηση που κάλεσε την readExpression (δηλαδή η main για την άσκηση Fractions).

Στο σημείο αυτό είναι η διαφορά των 2 εκδόσεων της readExperssion. Τα δύο παραδείγματα που ακολουθούν παρουσιάζουν την διαφορά αυτή.

5.1. 1η έκδοση της readExperssion

Η 1η έκδοση της readExperssion, η οποία έχει το παρακάτω function prototype,

```
void readExpression(char *operatorPtr, int *op1nPtr,int *op1dPtr,
                    int *op2nPtr,int *op2dPtr);
```

αποθηκεύει τα στοιχεία της έκφρασης που διαβάζει από τον χρήστη στις θέσεις μνήμης που δείχνουν τα πραγματικά ορίσματα της συνάρτησης, δηλαδή οι δείκτες operatorPtr, op1nPtr, κλπ. Βάζει δηλαδή τα δεδομένα που διαβάζει σε θέσεις μνήμης που έχει δεσμεύσει η συνάρτηση που κάλεσε την readExperssion.

Με τον τρόπο αυτό η συνάρτηση που κάλεσε την readExpression έχει πρόσβαση στα στοιχεία της έκφρασης και η readExpression δεν χρειάζεται να επιστρέψει κάτι (οπότε έχει ως επιστρεφόμενη τιμή void).

Αυτό σημαίνει πως η συνάρτηση που θα καλέσει την readExperssion θα πρέπει να έχει δηλώσει τις κατάλληλες μεταβλητές για την αποθήκευση των τιμών της έκφρασης και να περάσει στην readExperssion τους δείκτες αυτών των μεταβλητών.

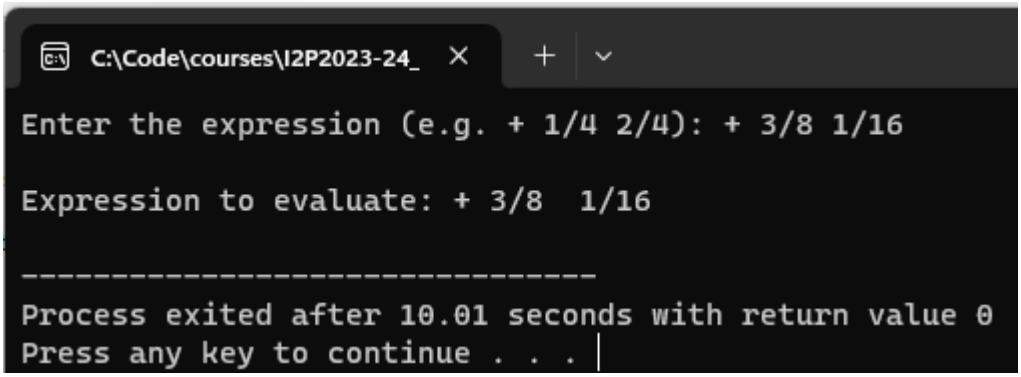
Παράδειγμα αξιοποίησης της συνάρτησης

Το Σχήμα 4 δίνει τον πηγαίο κώδικα από ένα παράδειγμα αξιοποίησης της συνάρτησης readExperssion. Προσέξτε την δήλωση των μεταβλητών για τα στοιχεία της έκφρασης ως γενικές μεταβλητές για να τι βλέπει και η displayExpression. Το Σχήμα 5 δίνει ένα screenshot από μία εκτέλεση του προγράμματος.

main.c main.c

```c
#include <stdio.h>
#include <stdlib.h>
#include "i2p.h"

void displayExpression();

char operator;
int op1ar,op1par,op2ar,op2par;

int main(int argc, char *argv[]) {

    readExpression(&operator,&op1ar,&op1par
    displayExpression();
    return 0;
}

void displayExpression(){
    printf("\nExpression to evaluate: %c %c
}
```

Σχήμα 4. Παράδειγμα κλήσης της readExpression της 1ης έκδοσης της i2p.



Σχήμα 5. Screenshot εκτέλεσης του προγράμματος κλήσης της readExpression (1ης έκδοσης της i2p).

5.2. 2η έκδοση της readExperssion

Η 2 έκδοση της readExperssion, η οποία έχει το παρακάτω function prototype,

```
Expression readExpression(void);
```

αποθηκεύει τα στοιχεία της έκφρασης που διαβάζει από τον χρήστη σε μία δική της μεταβλητή τύπου Expression και στη συνέχεια επιστρέφει αυτή την μεταβλητή στην συνάρτηση που την κάλεσε.

Αυτό σημαίνει πως η συνάρτηση που θα την καλέσει να πρέπει να έχει ορίσει μια μεταβλητή τύπου Expression και να καλέσει την `readExpression` για να της αποδώσει τιμή. Οι παρακάτω προτάσεις κάνουν αυτό ακριβώς.

```
Expression exp;
exp = readExpression();
```

Η πρώτη δηλώνει τη μεταβλητή exp ως τύπου `Expression` και η 2η καλεί την `readExpression` για να της αποδώσει τιμή.

Με τον τρόπο αυτό η συνάρτηση που κάλεσε την `readExpression` έχει πρόσβαση στα στοιχεία της έκφρασης καθώς αυτά βρίσκονται στην δική της μεταβλητή, δηλαδή την exp η οποία είναι τοπική μεταβλητή (local variable).

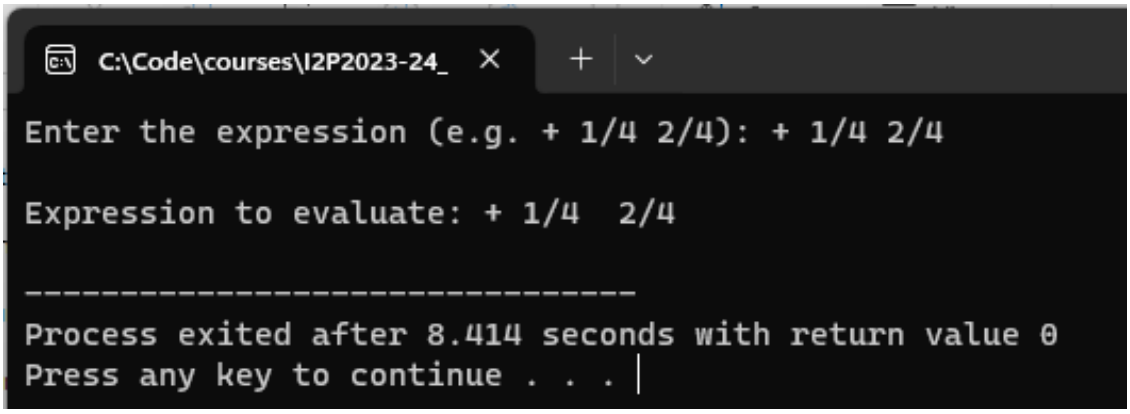
Παράδειγμα αξιοποίησης της συνάρτησης

Το Σχήμα 6 δίνει τον πηγαίο κώδικα από ένα παράδειγμα αξιοποίησης της συνάρτησης readExperssion. Προσέξτε τη δήλωση της μεταβλητής για τα στοιχεία της έκφρασης ως τοπικής μεταβλητής στην main οπότε δεν την βλέπει η displayExpression και αναγκαστικά την περνάμε ως όρισμα σε αυτήν κατά την κλήση της. Το Σχήμα 7 δίνει ένα screenshot από μία εκτέλεση του προγράμματος.

main.c

```c
#include <stdio.h>
#include <stdlib.h>
#include "i2p.h"

void displayExpression(Expression exp);

int main(int argc, char *argv[]) {
    Expression exp;

    exp=readExpression();
    displayExpression(exp);
}

void displayExpression(Expression exp){
    printf("\nExpression to evaluate: %c %
}
```

Σχήμα 6. Παράδειγμα κλήσης της readExpression της 2ης έκδοσης της i2p.



Σχήμα 7. Screenshot εκτέλεσης του προγράμματος κλήσης της readExpression (2ης έκδοσης της i2p).

5.3. Συνάρτηση ταξινόμησης πίνακα ακεραίων

Αν υποθέσουμε πως έχουμε τον παρακάτω πίνακα ακεραίων `ar` ο οποίος περιέχει 10 ακεραίους (int).

```
int ar[10];
```

Στην περίπτωση αυτή η κλήση της `sortInc4Int` διαμορφώνεται όπως παρακάτω:

```
sortInc4Int(ar, sizeof(ar)/sizeof(ar[0]), sizeof(ar[0]));
```

Τα πραγματικά ορίσματα περιγράφονται στη συνέχεια. Το `ar` είναι ο δείκτης στο πρώτο στοιχείο του πίνακα. Το `sizeof(ar)/sizeof(ar[0])` μας δίνει το πλήθος των στοιχείων του πίνακα και το `sizeof(ar[0])` μας δίνει το μέγεθος σε bytes του στοιχείου του πίνακα, δηλαδή στην περίπτωση μας του τύπου int.

5.4. Συνάρτηση ταξινόμησης πίνακα αλφαριθμητικών

Αν υποθέσουμε πως έχουμε τον παρακάτω πίνακα αλφαριθμητικών strAr ο οποίος περιέχει 100 αλφαριθμητικά μέγιστου πλάτους 40.

```
char strAr[100][40];
```

Στην περίπτωση αυτή η κλήση της `sortInc4String` διαμορφώνεται όπως παρακάτω:

```
sortInc4String((char *)strAr, sizeof(strAr)/sizeof(strAr[0]),
                sizeof(strAr[0]));
```

Τα πραγματικά ορίσματα περιγράφονται στη συνέχεια. Το `strAr` είναι ο δείκτης στο πρώτο στοιχείο του πίνακα. Το `sizeof(strAr)/sizeof(strAr[0])` μας δίνει το πλήθος των στοιχείων του πίνακα και το `sizeof(strAr[0])` μας δίνει το μέγεθος σε bytes του στοιχείου του πίνακα.