

Κρουστάλη Μαρία, e18084
3^η Εργαστηριακή Άσκηση
Πρωτόκολλα Διαδικτύου

ΠΕΡΙΕΧΟΜΕΝΑ

TCP Client	2
Αποστολή μηνύματος	2
Λήψη απάντησης	3
TCP Server	3
Λήψη μηνύματος	3
Αποστολή Απάντησης	4
Εφαρμογές	4
Περίπτωση 1 – Μεγάλοι Αριθμοί	5
Περίπτωση 2 – Μικροί Αριθμοί	5
Περίπτωση 3 – Μη έγκυροι αριθμοί εισόδου	6
Περίπτωση 4 – διαίρεση με 0	7

TCP CLIENT

ΑΠΟΣΤΟΛΗ ΜΗΝΥΜΑΤΟΣ

Αρχικά δίνουμε στον client την διεύθυνση IP του server και το port σύνδεσης με αυτόν. Έπειτα, δημιουργείται TCP socket για σύνδεση με τον server μέσω της IP και του port που δόθηκαν. Τα δεδομένα που θα σταλθούν μέσω του socket είναι:

- `message_type`, παίρνει τιμές 0 και 1 ανάλογα αν αποτελεί μήνυμα του client ή απάντηση από τον server.
- `message_id`, παίρνει οποιαδήποτε ακέραια τιμή η οποία περιέχεται και στο μήνυμα και στην απάντηση ώστε να αναγνωρίζεται ποια απάντηση αναφέρεται σε ποιο μήνυμα.
- `int1`, ο πρώτος ακέραιος αριθμός που εισάγει ο χρήστης.
- `int2`, ο δεύτερος ακέραιος αριθμός που επέλεξε ο χρήστης.
- `operator`, ειδικός χαρακτήρας που υποδηλώνει το είδος της πράξης.

Ορίζεται πίνακας με τα τέσσερα βασικά σύμβολα πράξεων και με for loop αποστέλλουμε στον server κάθε φορά τους αριθμούς με διαφορετικό σύμβολο πράξης.

Για κάθε ένα από τα τέσσερα μηνύματα που στέλνονται στον server υπολογίζουμε το μήκος του μηνύματος. Αναλυτικότερα:

- `message_type`, unsigned short, με μήκος 2 bytes
- `message_id`, unsigned int, μήκος 4 bytes
- `int1`, signed short, με μήκος 2 bytes
- `int2`, signed short, με μήκος 2 bytes
- `operator`, char, με μήκος 1 byte
- `message_length` (το μήκος του μηνύματος), unsigned short, με μήκος 2 bytes

Άρα το μήνυμα που θα σταλεί θα είναι μήκους 13 bytes.

Προκειμένου να στείλουμε τον operator, τύπου char, θα πρέπει να τον αποκωδικοποιήσουμε (utf-8) σε byte.

Στη συνέχεια, γίνεται η διαδικασία packing στα data που θα σταλθούν, καθορίζοντας για καθένα από τα παραπάνω δεδομένα το format που του αντιστοιχεί. Τελικά, το pack format είναι το `HHIhhx1s`:

- | | |
|----------------------------------|------------------------------|
| • H, <code>message_type</code> | • h, <code>int1</code> |
| • H, <code>message_length</code> | • h, <code>int2</code> |
| • I, <code>message_id</code> | • x1s, <code>operator</code> |

Τέλος, εκτυπώνεται η πράξη που θα εκτελεστεί (αριθμοί και τελεστής) και στέλνονται τα δεδομένα στον server.

ΛΗΨΗ ΑΠΑΝΤΗΣΗΣ

Όταν ο client λαμβάνει απάντηση, γνωρίζει ότι θα λάβει 12 bytes επειδή θα λάβει:

- `message_type`, ως response, unsigned short, 2 bytes
- `message_response_code`, ως τον κωδικό της απάντησης (0-3), unsigned short, 2 bytes
- `message_response_id`, κωδικός που καθορίζει σε ποιο μήνυμα αναφέρεται η απάντηση, int, 4 bytes
- `result`, αποτέλεσμα πράξης, float, 4 bytes.

Προκειμένου ο client να διαβάσει τα data πρέπει να κάνει την διαδικασία unpacking. Εδώ, χρησιμοποιείται το format *HHIf*, σύμφωνα με τους τύπους δεδομένων, όπως αναφέρθηκαν παραπάνω.

Στην συνέχεια, είναι απαραίτητο να ταυτοποιηθεί αν η απάντηση που λήφθηκε αναφέρεται στο συγκεκριμένο μήνυμα που στείλαμε, συγκρίνοντας τα `message_id` και `message_response_id`.

Έπειτα, ελέγχεται η τιμή του `message_response_code` όπου παίρνει τιμές από 0 έως 3:

- 0, για επιτυχή εκτέλεση πράξης. Το αποτέλεσμα εκτυπώνεται είτε ως ακέραιος (αν το δεκαδικό μέρος είναι 0, για τις πράξεις της πρόσθεσης, αφαίρεσης και πολλαπλασιασμού, είτε ως δεκαδικός, για την πράξη της διαίρεσης.
- 1, για μη αποδεκτό πρώτο αριθμό που δίνει ο χρήστης, δηλαδή εκτός του εύρους 0 - 30000
- 2, για μη αποδεκτό δεύτερο αριθμό που δίνει ο χρήστης, εκτός του εύρους 0 - 30000
- 3, για τιμή 0 στον δεύτερο αριθμό, καθώς δεν μπορεί να πραγματοποιηθεί διαίρεση με το 0.

Αφού εκτυπωθεί ένα από τα παραπάνω μηνύματα ο client κλείνει το socket, δηλαδή τερματίζει την επικοινωνία του με τον server.

TCP SERVER

ΛΗΨΗ ΜΗΝΥΜΑΤΟΣ

Στον κώδικα του server, αρχικά, καθορίζεται η IP διεύθυνση του και το port στο οποίο θα δέχεται συνδέσεις. Με τα παραπάνω στοιχεία δημιουργείται το server socket στο οποίο «ακούει» για αιτήματα συνδέσεων από clients.

Όταν δεχτεί κάποιο αίτημα, λαμβάνει την IP διεύθυνση του client και δέχεται σταδιακά τα data που του αποστέλλονται. Συγκεκριμένα, χρησιμοποιείται ένα loop ώστε ο client και ο server να μπορούν να ανταλλάσσουν δεδομένα στο ίδιο socket. Δηλαδή, χωρίς να χρησιμοποιείται νέο socket για κάθε πράξη.

Αρχικά, ο server δέχεται 4 bytes:

- `message_type`, με τιμή 0 επειδή αποτελεί μήνυμα από τον client
- `message_length`, με το συνολικό μήκος τους μηνύματος

Τα δύο παραπάνω είναι τύπου `unsigned short` άρα έχουν μήκος 2 bytes το καθένα. Γίνεται διαδικασία `unpacking` σε αυτά με `format HH`, λόγω του τύπου τους, `unsigned short`.

Στην συνέχεια, λαμβάνονται και τα υπόλοιπα bytes του μηνύματος, δεδομένου ότι γνωρίζουμε το συνολικό μήκος του (`message_length`). Έτσι, περνούν στον server και όλα τα υπόλοιπα δεδομένα:

- `message_id`, id κοινό στο μήνυμα και την αντίστοιχη απάντησή του
- `int1`, πρώτος ακέραιος που διάλεξε ο χρήστης
- `int2`, δεύτερος ακέραιος που διάλεξε ο χρήστης
- `operator`, ο τελεστής κάθε πράξης που στέλνεται από τον client

Ανάλογα με τον κάθε τελεστή εκτελείται η αντίστοιχη πράξη και εκχωρείται σε μεταβλητή `result`. Επιπλέον, ορίζεται το `message_type` σε 1, δηλαδή ως απάντηση και το `message_response_code`. Για το τελευταίο οι τιμές είναι 0 έως 3 ανάλογα:

- 0, αν η πράξη εκτελείται με επιτυχία
- 1 ή 2, αν κάποιος από τους αριθμούς είναι εκτός εύρους
- 3, αν ο 2^{ος} αριθμός που διαιρεί είναι 0

Προκειμένου να σταλθεί το αποτέλεσμα με τη σωστή μορφή, μετατρέπεται σε τύπο `float`, ώστε να υπάρχει ακρίβεια δεκαδικών ψηφίων για τη πράξη της διαίρεσης.

ΑΠΟΣΤΟΛΗ ΑΠΑΝΤΗΣΗΣ

Τέλος, γίνεται αποστολή του αποτελέσματος στον client μαζί με τα `message_type`, `message_response_code` και `message_id`. Στο `packing` αυτών χρησιμοποιείται το `format HHIf`, όπου:

- H, `message_type`, `unsigned short`
- H, `message_response_code`, `unsigned short`
- I, `message_id`, `int`
- F, `result`, `float`

Εκτυπώνεται στον server πιθανό λάθος που μπορεί να έγινε κατά την πράξη και κλείνει το `socket`.

ΕΦΑΡΜΟΓΕΣ


Παρακάτω, παρουσιάζονται παραδείγματα εκτέλεσης της παραπάνω διαδικασίας με διάφορους αριθμούς ως είσοδο από τον χρήστη.

Στον client ζητούνται οι αριθμοί από τον χρήστη, έπειτα, παρουσιάζεται η κάθε πράξη και, τέλος, το αποτέλεσμα. Όταν εκτελεστούν και οι τέσσερις πράξεις το `socket` κλείνει.

Στον server παρουσιάζονται τα στοιχεία (IP, port) του server και του client με τον οποίο επικοινωνεί. Εκτυπώνεται κάθε πράξη που λαμβάνει καθώς και το αποτέλεσμα της. Επιπλέον, επιστρέφεται οποιοδήποτε error μπορεί να δημιουργήθηκε κατά την εκτέλεση πράξης.

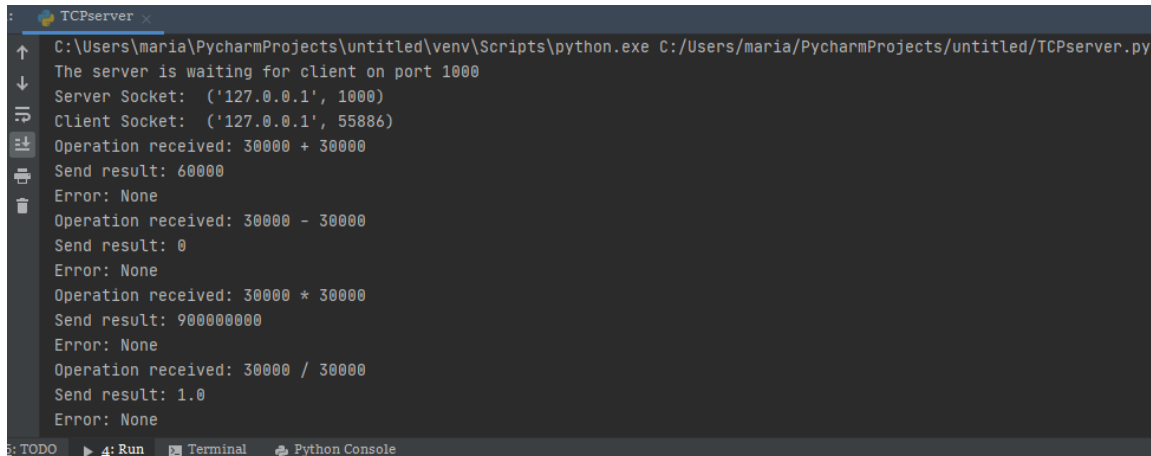
ΠΕΡΙΠΤΩΣΗ 1 – ΜΕΓΑΛΟΙ ΑΡΙΘΜΟΙ

Για είσοδο 30000 και 30000 τα αποτελέσματα των πράξεων στον client θα είναι μέγιστα, όπως φαίνεται παρακάτω:



```
Terminal: Local x +
(venv) C:\Users\maria\PycharmProjects\untitled>python tcpclient.py
Insert first integer (0-30000):
30000
Insert second integer (0-30000):
30000
Operation sent: 30000 + 30000
Result: 60000
Operation sent: 30000 - 30000
Result: 0
Operation sent: 30000 * 30000
Result: 900000000
Operation sent: 30000 / 30000
Result: 1
★ Operation completed. Closing socket...
```

Στον server:



```
TCPserver x
C:\Users\maria\PycharmProjects\untitled\venv\Scripts\python.exe C:/Users/maria/PycharmProjects/untitled/TCPserver.py
The server is waiting for client on port 1000
Server Socket: ('127.0.0.1', 1000)
Client Socket: ('127.0.0.1', 55886)
Operation received: 30000 + 30000
Send result: 60000
Error: None
Operation received: 30000 - 30000
Send result: 0
Error: None
Operation received: 30000 * 30000
Send result: 900000000
Error: None
Operation received: 30000 / 30000
Send result: 1.0
Error: None
```

ΠΕΡΙΠΤΩΣΗ 2 – ΜΙΚΡΟΙ ΑΡΙΘΜΟΙ

Για τους αριθμούς 1 και 2, τα αποτελέσματα στον Client:

```
Terminal: Local x +
(venv) C:\Users\maria\PycharmProjects\untitled>python TCPClient.py
Insert first integer (0-30000):
1
Insert second integer (0-30000):
2
Operation sent: 1 + 2
Result: 3
Operation sent: 1 - 2
Result: -1
Operation sent: 1 * 2
Result: 2
Operation sent: 1 / 2
Result: 0.5
Operation completed. Closing socket...
(venv) C:\Users\maria\PycharmProjects\untitled>
```

Στον server:

```
TCPserver x
C:\Users\maria\PycharmProjects\untitled\venv\Scripts\python.exe C:/Users/maria/PycharmProjects/untitled/TCPserver.py
The server is waiting for client on port 1000
Server Socket: ('127.0.0.1', 1000)
Client Socket: ('127.0.0.1', 61590)
Operation received: 1 + 2
Send result: 3
Error: None
Operation received: 1 - 2
Send result: -1
Error: None
Operation received: 1 * 2
Send result: 2
Error: None
Operation received: 1 / 2
Send result: 0.5
Error: None
```

ΠΕΡΙΠΤΩΣΗ 3 – ΜΗ ΕΓΚΥΡΟΙ ΑΡΙΘΜΟΙ ΕΙΣΟΔΟΥ

Αρχικά δίνονται οι αριθμοί 30010 και 8, όπου ο πρώτος είναι εκτός του εύρους 0 – 30000:

```
Terminal: Local x +
(venv) C:\Users\maria\PycharmProjects\untitled>python TCPClient.py
Insert first integer (0-30000):
30010
Insert second integer (0-30000):
8
Operation sent: 30010 + 8
Error: First integer is not 0-30000!
Operation sent: 30010 - 8
Error: First integer is not 0-30000!
Operation sent: 30010 * 8
Error: First integer is not 0-30000!
Operation sent: 30010 / 8
Error: First integer is not 0-30000!
Operation completed. Closing socket...
(venv) C:\Users\maria\PycharmProjects\untitled>
```

Έπειτα, δίνονται οι αριθμοί 5004 και 30001, όπου ο δεύτερος είναι εκτός εύρους 0 – 30000:

```
Terminal: Local x +
(venv) C:\Users\maria\PycharmProjects\untitled>python TCPClient.py
Insert first integer (0-30000):
5004
Insert second integer (0-30000):
30001
Operation sent: 5004 + 30001
Error: Second integer is not 0-30000!
Operation sent: 5004 - 30001
Error: Second integer is not 0-30000!
Operation sent: 5004 * 30001
Error: Second integer is not 0-30000!
Operation sent: 5004 / 30001
Error: Second integer is not 0-30000!

Operation completed. Closing socket...
★ (venv) C:\Users\maria\PycharmProjects\untitled>
```

Και στις δύο περιπτώσεις, σε κάθε πράξη, επιστρέφεται το κατάλληλο μήνυμα.

ΠΕΡΙΠΤΩΣΗ 4 – ΔΙΑΙΡΕΣΗ ΜΕ 0

Δίνονται οι αριθμοί 7896 και 0, όπου ο δεύτερος είναι εντός του εύρους αλλά η πράξη δεν είναι επιτρεπτή. Έτσι, θα εκτελεστούν οι υπόλοιπες πράξεις, πρόσθεση, αφαίρεση, πολλαπλασιασμός, αλλά στην διαίρεση θα επιστραφεί κατάλληλο μήνυμα.

```
Terminal: Local x +
(venv) C:\Users\maria\PycharmProjects\untitled>python TCPClient.py
Insert first integer (0-30000):
7896
Insert second integer (0-30000):
0
Operation sent: 7896 + 0
Result: 7896
Operation sent: 7896 - 0
Result: 7896
Operation sent: 7896 * 0
Result: 0
Operation sent: 7896 / 0
Error: Operation division, second integer must not be 0!

Operation completed. Closing socket...
★
```