C: Από τη Θεωρία στην Εφαρμογή

Κεφάλαιο 3ο
Είσοδος Δεδομένων

Γ. Σ. Τσελίκης – Ν. Δ. Τσελίκας

Η συνάρτηση `scanf()`

- Η συνάρτηση `scanf()` χρησιμοποιείται για την είσοδο δεδομένων από ένα αρχείο εισόδου, το οποίο ονομάζεται `stdin` *(standard input stream)* και εξ' ορισμού συνδέεται με το πληκτρολόγιο

- Η `scanf()` δέχεται μία μεταβλητή λίστα παραμέτρων, παρόμοια με την `printf()`, δηλαδή:

  - Η πρώτη παράμετρος είναι ένα **αλφαριθμητικό μορφοποίησης *(format string)***, το οποίο, συνήθως, περιέχει μόνο απλά προσδιοριστικά μετατροπής (π.χ. `%d` για μεταβλητές τύπου `int`, `%f` για μεταβλητές τύπου `float` κτλ...)

  - Οι επόμενες προαιρετικές παράμετροι είναι οι **διευθύνσεις μνήμης** των μεταβλητών στις οποίες θα εκχωρηθούν τα δεδομένα που θα εισάγει ο χρήστης από το πληκτρολόγιο

Κάθε προσδιοριστικό μετατροπής πρέπει να αντιστοιχεί σε μία διεύθυνση μεταβλητής και η αντιστοίχιση γίνεται ένα προς ένα

Για μεταβλητές τύπου `double`, χρησιμοποιείται το προσδιοριστικό μετατροπής `%lf` και όχι το `%f`, το οποίο χρησιμοποιείται μόνο για μεταβλητές τύπου `float`

Παράδειγμα 1

- Π.χ.

```
int i;
scanf("%d", &i);
```

- Ο χαρακτήρας **&**, που μπαίνει πριν από το όνομα της μεταβλητής, ονομάζεται **τελεστής διεύθυνσης** και χρησιμοποιείται για να αποθηκευτεί ο αριθμός που θα εισάγει ο χρήστης στη διεύθυνση μνήμης της μεταβλητής **i**

- Περισσότερες λεπτομέρειες για τη σημασία του τελεστή διεύθυνσης **&** θα δούμε αναλυτικά στους "δείκτες της C"

Παράδειγμα 2 (Ι)

- Μπορούμε να χρησιμοποιήσουμε τη συνάρτηση `scanf()` για να διαβάσουμε περισσότερες από μία τιμές από το πληκτρολόγιο και να τις αποθηκεύσουμε ως τιμές σε κάποιες μεταβλητές του προγράμματος
- Π.χ.

```
int i;
float j;
scanf("%d%f", &i, &j);
```

- Η πρώτη παράμετρος της `scanf()` είναι το αλφαριθμητικό μορφοποίησης `%d%f`, ενώ οι επόμενες παράμετροι είναι οι διευθύνσεις μνήμης των μεταβλητών `i` και `j` αντίστοιχα
  - Το `%d` αντιστοιχεί στη διεύθυνση της μεταβλητής `i`
  - Το `%f` αντιστοιχεί στη διεύθυνση της μεταβλητής `j`
  - Δηλαδή η αντιστοίχιση γίνεται ένα προς ένα και από αριστερά προς τα δεξιά
- Για την είσοδο των δεδομένων χρησιμοποιείται συνήθως το «κενό διάστημα» (space) μεταξύ των διαφορετικών τιμών που εισάγονται, δεδομένου ότι κατά το διάβασμα αριθμητικών τιμών, η `scanf()` αγνοεί όλα τα λευκά διαστήματα (π.χ. κενά διαστήματα, tab, χαρακτήρα νέας γραμμής) που μπορεί να υπάρχουν πριν από κάθε αριθμητική τιμή

Παράδειγμα 2 (ΙΙ)

- Στο προηγούμενο παράδειγμα, αν ο χρήστης εισάγει π.χ. τις τιμές `10` και `4.65`, αυτές θα πρέπει να απέχουν μεταξύ τους ένα ή περισσότερα κενά
- Για να ληφθούν από τη συνάρτηση `scanf()` πρέπει μετά ο χρήστης να πατήσει Enter
- Τότε, η τιμή `10` αποθηκεύεται στη μεταβλητή `i` και η τιμή `4.65` στη μεταβλητή `j`, αντίστοιχα

Παραδείγματα 3 & 4

- Στο επόμενο παράδειγμα, η `scanf()` διαβάζει έναν χαρακτήρα και τον αποθηκεύει στη μεταβλητή `ch`

```
char ch;
scanf("%c", &ch);
```

- Στο επόμενο παράδειγμα, η `scanf()` διαβάζει ένα αλφαριθμητικό και το αποθηκεύει στον πίνακα χαρακτήρων `str`
- Παρατηρήστε, ότι πριν από τη μεταβλητή `str` δεν χρησιμοποιείται ο τελεστής `&`, γιατί – όπως θα δούμε στο κεφάλαιο των "Δεικτών"– το όνομα ενός πίνακα ισοδυναμεί με τη διεύθυνση του πρώτου στοιχείου του

```
char str[100];
scanf("%s", str);
```

- Αν ο χρήστης εισάγει το αλφαριθμητικό `sample` και πατήσει Enter, τότε οι χαρακτήρες του θα αποθηκευτούν στις αντίστοιχες θέσεις του πίνακα `str`
- Δηλαδή, η τιμή του `str[0]` θα γίνει `'s'`, του `str[1]` θα γίνει `'a'`, του `str[2]` θα γίνει `'m'`, κ.ο.κ.
- Το παράδειγμα αυτό θα το κατανοήσετε καλύτερα αργότερα, αφού θα μιλήσουμε για πίνακες, χαρακτήρες και αλφαριθμητικά

Παρατηρήσεις

Να θυμάστε ότι η `scanf()` απαιτεί τον τελεστή διεύθυνσης `&` πριν από το όνομα κάθε αριθμητικής μεταβλητής (π.χ. `int`, `double`, `char`, `float`, …)

Αν τον ξεχάσετε, το πρόγραμμά σας δεν θα εκτελεστεί σωστά

Αντίθετα, όταν ο τύπος της μεταβλητής είναι δείκτης, ο τελεστής διεύθυνσης δεν χρειάζεται

Για να διαβάσετε με τη `scanf()` ένα αλφαριθμητικό που μπορεί να αποτελείται από πολλές λέξεις (π.χ. `"Text with multiple words"`), πρέπει να χρησιμοποιήσετε μία πιο σύνθετη μορφή της,

π.χ.: `scanf("%[^\n]",str);`

γιατί η `scanf()` εξ' ορισμού σταματάει το διάβασμα του αλφαριθμητικού όταν συναντήσει έναν κενό χαρακτήρα

Τί επιστρέφει η συνάρτηση scanf() ???

- Η συνάρτηση **scanf()** επιστρέφει έναν ακέραιο αριθμό που δηλώνει πόσα από τα δεδομένα εισόδου διαβάστηκαν και εκχωρήθηκαν στις μεταβλητές του προγράμματος, ενώ οι τιμές που δεν διαβάστηκαν παραμένουν στο **stdin**

- Π.χ. στο παράδειγμα:

```
int i;
scanf("%d", &i);
```

αν ο χρήστης εισάγει έναν ακέραιο, η συνάρτηση **scanf()** επιστρέφει την τιμή **1**

- Ενώ στο παράδειγμα:

```
int i;
float j;
scanf("%d%f", &i, &j);
```

αν ο χρήστης εισάγει έναν ακέραιο και έναν πραγματικό αριθμό, η συνάρτηση **scanf()** επιστρέφει την τιμή **2**

Παρατηρήσεις

- Η συνάρτηση **scanf()** δεν είναι μία ασφαλής συνάρτηση και πρέπει να τη χρησιμοποιείτε με μεγάλη προσοχή
  - Σε επαγγελματικές εφαρμογές πρέπει να ελέγχετε την τιμή επιστροφής της

```
#include <stdio.h>
int main(void)
{
	int num;
	printf("Enter number: ");
	while(scanf("%d", &num) != 1)
	{
		printf("Enter number: ");
		while(getchar() != '\n'); /* Καθαρισμός μνήμης πληκτρολογίου. */
	}
	printf("Inserted value: %d\n", num);
	return 0;
}
```

  - Σχεδόν σε όλα τα προγράμματα που θα υλοποιήσουμε, όταν χρησιμοποιούμε τη συνάρτηση **scanf()**, για λόγους απλότητας, δεν θα ελέγχουμε την τιμή επιστροφής της και θα θεωρούμε ότι οι τιμές που εισάγει ο χρήστης θα είναι σε συμφωνία με τα προσδιοριστικά μετατροπής

Παρεμβολή απλών χαρακτήρων στη scanf() (I)

- Στην πιο συνηθισμένη χρήση της, το αλφαριθμητικό μορφοποίησης της `scanf()` δεν περιέχει απλούς χαρακτήρες παρά μόνο τα προσδιοριστικά μετατροπής (π.χ. `%d`, `%f`,…)

- Ωστόσο, αν παρεμβληθούν κάποιοι απλοί χαρακτήρες, τότε πρέπει οι αντίστοιχοι χαρακτήρες να εισαχθούν και από το πληκτρολόγιο

- Π.χ. στην επόμενη `scanf()` παρεμβάλλεται ο χαρακτήρας κόμμα (,)

```
#include <stdio.h>
int main(void)
{
        int a, b;
        scanf("%d , %d", &a, &b);
        printf("%d %d\n", a, b);
        return 0;
}
```

- Για να λειτουργήσει σωστά αυτό το πρόγραμμα πρέπει οι ακέραιες τιμές που θα εισάγει ο χρήστης να διαχωρίζονται μεταξύ τους με κόμμα (,)

Παρεμβολή απλών χαρακτήρων στη scanf() (II)

- Αν αντί για κόμμα (όπως είδαμε στο προηγούμενο παράδειγμα), μεταξύ των ειδικών χαρακτήρων μέσα στην scanf() χρησιμοποιούνταν ο χαρακτήρας 'm', (όπως φαίνεται παρακάτω), θα έπρεπε να εισαχθεί ο χαρακτήρας 'm' μεταξύ των τιμών που θα έδινε ο χρήστης από το πληκτρολόγιο

```
#include <stdio.h>
int main(void)
{
        int a, b;
        scanf("%dm%d", &a, &b);
        printf("%d %d\n", a, b);
        return 0;
}
```

- Αν δηλαδή ο χρήστης επιθυμούσε να εισάγει τις τιμές 12 και 43, θα έπρεπε να πληκτρολογήσει: 12m43

Γενικά, προτείνουμε <u>να μην παρεμβάλλεται</u> κάποιος χαρακτήρας μεταξύ των προσδιοριστικών μετατροπής, έτσι ώστε να μην χρειάζεται κάποια επιπλέον εισαγωγή χαρακτήρων από τον χρήστη (σκεφτείτε επίσης ότι ο τελικός χρήστης του προγράμματος δεν είναι απαραίτητα και ο δημιουργός του προγράμματος)

Καθαρισμός Μνήμης

- Η συνάρτηση `scanf()` δεν θα λειτουργήσει σωστά, αν ο χρήστης δεν εισάγει τα δεδομένα σύμφωνα με την ακολουθία των προσδιοριστικών μετατροπής που ορίζονται σε αυτήν

- Στο επόμενο παράδειγμα η `scanf()` αναμένει μία ακέραια και μία πραγματική τιμή

```
#include <stdio.h>
int main(void)
{
        int i;
        double j;

        printf("Enter numbers: ");

        scanf("%d", &i);
        scanf("%lf", &j);

        printf("Num1 = %d, Num2 = %f\n", i, j);
        return 0;
}
```

- Τι θα συμβεί αν ο χρήστης εισαγάγει – έστω κατά λάθος – ως ακέραια τιμή την τιμή `5.65` ?

Παράδειγμα (Ι)

- Η συνάρτηση **scanf()** δεν διαβάζει τον χαρακτήρα νέας γραμμής που πληκτρολογεί ο χρήστης στο τέλος της εισαγωγής δεδομένων
- Αυτός ο χαρακτήρας θα διαβαστεί στην επόμενη κλήση της **scanf()**
- Αν όμως, στην επόμενη κλήση της, η **scanf()** χρησιμοποιείται για το διάβασμα χαρακτήρων, τότε θα διαβαστεί μόνο αυτός ο χαρακτήρας (της νέας γραμμής) και οι υπόλοιποι θα αγνοηθούν
- Π.χ. το επόμενο πρόγραμμα δεν θα εκτελεστεί σωστά

```
#include <stdio.h>
int main(void)
{
        char ch;
        int i;

        printf("Enter number: ");
        scanf("%d",&i);

        printf("Enter character: ");
        scanf("%c",&ch);

        printf("Int = %d and Char = %c\n",i,ch);
        return 0;
}
```

Παράδειγμα (ΙΙ)

- Αν αντιστρέψουμε στο προηγούμενο παράδειγμα τη σειρά του διαβάσματος, τότε το πρόγραμμα θα εκτελεστεί σωστά, αφού – σύμφωνα με προηγούμενη παρατήρηση – ο χαρακτήρας νέας γραμμής που υπάρχει πριν από την αριθμητική τιμή αγνοείται

```
#include <stdio.h>
int main(void)
{
        char ch;
        int i;

        printf("Enter character: ");
        scanf("%c",&ch);

        printf("Enter number: ");
        scanf("%d",&i);

        printf("Int = %d and Char = %c\n",i,ch);
        return 0;
}
```

Τρόποι Καθαρισμού Μνήμης

- Ένας τρόπος για να αδειάσουμε τη μνήμη του πληκτρολογίου από τα δεδομένα που έχουν παραμείνει είναι με τη χρήση της συνάρτησης `getchar()` χρησιμοποιώντας τον παρακάτω επαναληπτικό βρόχο

```
while(getchar() != '\n');
```

ή – ακόμα καλύτερα – τον παρακάτω:

```
while((ch = getchar()) != '\n' && ch != EOF);
```

- Ωστόσο, πολλοί προγραμματιστές χρησιμοποιούν τη συνάρτηση `fflush()`

```
fflush(stdin);
```

Προσέξτε όμως, σύμφωνα με το πρότυπο της C, η συμπεριφορά της `fflush()` όταν χρησιμοποιείται με όρισμα το `stdin` **είναι ακαθόριστη** πράγμα που σημαίνει, **ΜΗΝ ΤΗ ΧΡΗΣΙΜΟΠΟΙΕΙΤΕ**

Παραδείγματα (1)

- Γράψτε ένα πρόγραμμα που να διαβάζει την ακτίνα ενός κύκλου και να εμφανίζει το εμβαδό του αντίστοιχου κύκλου και την περίμετρό του

```
#include <stdio.h>

#define PI 3.14159

int main(void)
{
        double radius;

        printf("Enter radius: ");
        scanf("%lf", &radius);
        printf("%f %f\n", PI*radius*radius, 2*PI*radius);
        return 0;
}
```

Παραδείγματα (2)

- Γράψτε ένα πρόγραμμα, το οποίο να διαβάζει δύο ακέραιες τιμές, να τις αποθηκεύει σε δύο ακέραιες μεταβλητές και να αντιμεταθέτει τις τιμές τους

```
#include <stdio.h>
int main(void)
{
        int i, j, temp;

        printf("Enter numbers: ");
        scanf("%d%d", &i, &j);

        temp = i;
        i = j;
        j = temp;
        printf("%d %d\n", i, j);
        return 0;
}
```