

Διάλεξη 8 - Ροή Ελέγχου #2

Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

Εισαγωγή στον Προγραμματισμό

Θανάσης Αυγερινός / Τάκης Σταματόπουλος

Ανακοινώσεις / Διευκρινήσεις

- Έτρεξα τον παρακάτω πολλαπλασιασμό:

```
for(prod = 1, i = 105 ; i <= 999 ; i += 14) {  
    prod *= i;  
  
}
```

και μου τύπωσε το παρακάτω - τι συμβαίνει;

```
$ ./prod
```

```
-1187656959
```

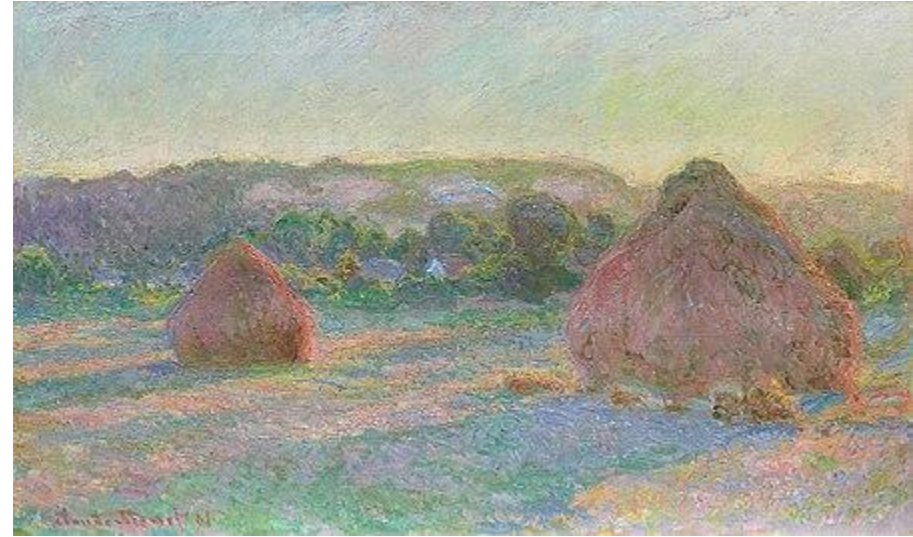
Την προηγούμενη φορά

Επίλυση Προβλημάτων



Σήμερα

- Παραδείγματα Επαναληπτικών
Δομών
- Άλλες Δομές Ελέγχου
- Επίλυση Προβλημάτων



Θέλω να βρω το γινόμενο όλων των τριψήφιων περιττών που διαιρούνται με το 7. Πως;

Ένα for loop με μια μεταβλητή που αυξάνεται και έλεγχο για $\text{mod } 7 = 0$:

```
for(product = 1, i = 100 ; i <= 999 ; i +=2) {  
    if ( i % 7 == 0 )  
        product *= i;  
}
```

Ή

```
for(prod = 1, i = 105 ; i <= 999 ; i += 14) {  
    prod *= i;  
}
```

Θέλω να τυπώσω (αν υπάρχει) τον μικρότερο τριψήφιο που είναι πολλαπλάσιο του 2 και του 5 αλλά όχι του 4. Πως;

Θέλω να τυπώσω (αν υπάρχει) τον μικρότερο τριψήφιο που είναι πολλαπλάσιο του 2 και του 5 αλλά όχι του 4. Πως;

```
for(i = 100 ; i < 1000 ; i++) {  
    if (i % 2 == 0 && i % 5 == 0 && i % 4 != 0) {  
        printf("%d\n", i);  
    }  
}
```

Υπάρχει κάποιο θέμα με αυτήν την υλοποίηση;

Θέλω να τυπώσω (αν υπάρχει) τον μικρότερο τριψήφιο που είναι πολλαπλάσιο του 2 και του 5 αλλά όχι του 4. Πως;

```
int found = 0;
for(i = 100 ; i < 1000 && !found; i++) {
    if (i % 2 == 0 && i % 5 == 0 && i % 4 != 0) {
        printf("%d\n", i);
        found = 1;
    }
}
```

Χρησιμοποιούμε μια μεταβλητή found για να ελέγχουμε πότε θέλουμε να "σπάσουμε" το loop και να συνεχίσουμε το πρόγραμμά μας. Η C μας προσφέρει και εναλλακτικό τρόπο έκφρασης για να επιτύχουμε το ίδιο αποτέλεσμα.

Εντολή **break** (break Statement)

Η εντολή **break** μας επιτρέπει να "σπάσουμε" τον τρέχοντα βρόχο τερματίζοντας τον μετά την εκτέλεσή της.

```
while ( ... ) {
```

```
    break;
```

```
}
```



Η ίδια εντολή μας επιτρέπει να "σπάσουμε" και τις άλλες δομές ελέγχου όπως είναι το **do-while** και το **for**.

Παράδειγμα με break


Εύρεση του μικρότερου τριψήφιου που είναι πολλαπλάσιο του 2 και του 5 αλλά όχι του 4 με break:

```
for(i = 100 ; i < 1000; i++) {  
    if (i % 2 == 0 && i % 5 == 0 && i % 4 != 0) {  
        printf("%d\n", i);  
        break;  
    }  
}
```

Εντολή continue (continue Statement)

Η εντολή **continue** μας επιτρέπει να διακόψουμε την τρέχουσα επανάληψη του βρόχου και να συνεχίσουμε στην επόμενη.

```
while ( ... ) {  
    continue;  
    ...  
}
```



Η ίδια εντολή μας επιτρέπει να διακόψουμε την τρέχουσα επανάληψη και στις άλλες δομές ελέγχου όπως είναι το **do-while** και το **for**.

Παράδειγμα με continue

Τύπωσε όλους τους τριψήφιους εκτός από τα πολλαπλάσια του 5:

```
for(i = 100 ; i < 1000; i++) {  
    if (i % 5 == 0) {  
        continue;  
    }  
    printf("%d\n", i);  
}
```

Θέλω να τυπώσω το αγγλικό όνομα κάθε ψηφίου - Πως;

```
if (number == 0)
    printf("zero\n");
else if (number == 1)
    printf("one\n");
else if (number == 2)
    printf("two\n");
...
else if (number == 9)
    printf("nine\n");
else
    printf("unknown\n");
```

Εντολή switch (switch Statement)

Η εντολή ελέγχου **switch** είναι εναλλακτική της **if-else-if** δομής, όταν χρειάζεται να ελέγξουμε μία έκφραση για τις δυνατές τιμές που μπορεί να πάρει και να χειριστούμε την κάθε περίπτωση με διαφορετικό τρόπο.

```
switch (έκφραση) {  
    case σταθερά1:  
    case σταθερά2:  
    ...  
    default:  
}
```


Θέλω να τυπώσω το αγγλικό όνομα κάθε ψηφίου - Πως;

```
switch (number) {  
    case 0:  
        printf("zero\n");  
        break;  
    case 1:  
        printf("one\n");  
        break;  
    ...  
    case 9:  
        printf("nine\n");  
        break;  
    default:  
        printf("unknown\n");  
        break;  
}
```

Θέλω να τυπώσω το αγγλικό όνομα κάθε ψηφίου - Πως;

Τα break
ολοκληρώνουν
την εκτέλεση

```
switch (number) {  
    case 0:  
        printf("zero\n");  
        break;  
    case 1:  
        printf("one\n");  
        break;  
    ...  
    case 9:  
        printf("nine\n");  
        break;  
    default:  
        printf("unknown\n");  
        break;  
}
```



Θέλω να τυπώσω την εποχή κάθε μήνα - Πως;

Μπορούμε να
διαχειριστούμε
πολλά cases μαζί

```
switch (month) {  
    case 12:  
    case 1:  
    case 2:  
        printf("winter\n");  
        break;  
    case 3:  
    case 4:  
    case 5:  
        printf("spring\n");  
        break;  
    case 6:  
    case 7:  
    case 8:  
        printf("summer\n");  
        break;  
    ...  
}
```

Θέλω να τυπώσω την εποχή κάθε μήνα - Πως;

Σε κάθε case
ελέγχεται ισότητα ==
με την τιμή δεν
μπορούμε να
γράψουμε άλλες
λογικές συνθήκες
(πχ >, <, κτλ)

```
switch (month) {  
    case 12:  
    case 1:  
    case 2:  
        printf("winter\n");  
        break;  
    case 3:  
    case 4:  
    case 5:  
        printf("spring\n");  
        break;  
    case 6:  
    case 7:  
    case 8:  
        printf("summer\n");  
        break;  
    ...  
}
```

Κάθε έκφραση που
χρησιμοποιούμε
στο switch και
σταθερά που
χρησιμοποιούμε
στο case πρέπει να
είναι ακέραιος

Εντολή goto (goto Statement)

Η εντολή **goto** μεταφέρει την εκτέλεση του προγράμματος σε άλλη εντολή της ίδιας συνάρτησης με την προϋπόθεση ότι η εντολή έχει μία ετικέτα (label). Γενικής μορφής:

goto label;

Η σύνταξη για την ετικέτα είναι η ακόλουθη:

label:

Παράδειγμα goto

Εύρεση του μικρότερου τριψήφιου που είναι πολλαπλάσιο του 2 και του 5 αλλά όχι του 4 με goto:

```
for(i = 100 ; i < 1000; i++) {  
    if (i % 2 == 0 && i % 5 == 0 && i % 4 != 0) {  
        printf("%d\n", i);  
        goto exit_for;  
    }  
}  
exit_for:
```

Δομημένος προγραμματισμός σημαίνει ΟΧΙ goto

Η χρήση goto συχνά οδηγεί σε δυσνόητο κώδικα (χρήσιμο για [obfuscation contests](#) και [spaghetti code](#)!) και καλό είναι να αποφεύγεται. Μπορεί κανείς να γράψει εκατομμύρια γραμμές κώδικα χωρίς να την χρειαστεί.

Edgar Dijkstra: Go To Statement Considered Harmful

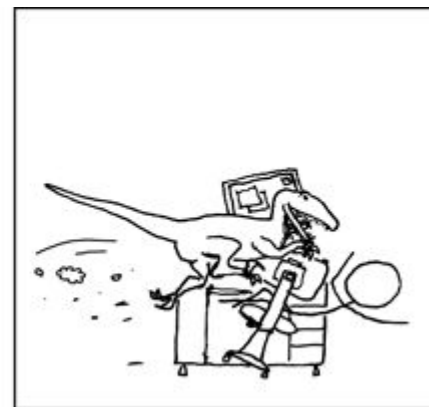
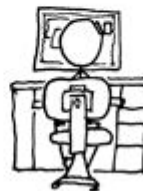
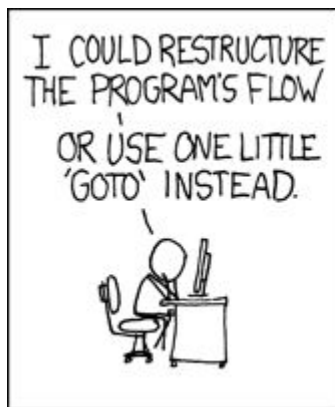
Go To Statement Considered Harmful

Key Words and Phrases: go to statement, jump instruction, branch instruction, conditional clause, alternative clause, repetitive clause, program intelligibility, program sequencing

dynamic progress is only characterized by the call of the procedure we refer to. we can characterize the progress of a program by textual indices, the length of the dynamic depth of procedure calls.

Να χρησιμοποιήσω goto?

ΌΧΙ



Συζήτηση για εργασία / README

Για την Επόμενη Φορά

- Από τις σημειώσεις του κ. Σταματόπουλου συνιστώ να έχετε καλύψει τα πάντα μέχρι την σελίδα 71.
- [Break and Continue](#)
- [Switch statement](#)
- [Considered harmful](#)

Ευχαριστώ και καλή μέρα εύχομαι!
Keep Coding ;)