Εντολές επιλογής - επανάληψης

Εντολές επιλογής

Η εντολή if

- Η βασική εντολή με την οποία μπορεί να γίνει επιλογή στο αν θα εκτελεστεί κάποιος κώδικας είναι η if
- Σύνταξη της if

if (condition) statement;

- ΠΡΟΣΟΧΗ
  - το condition πρέπει να είναι ΠΑΝΤΑ μέσα σε παρενθέσεις
  - Η if ισχύει για το επόμενο statement. Αν εμείς θέλουμε να περιλαμβάνει παραπάνω από μία εντολές τότε πρέπει να χρησιμοποιήσουμε ένα compound statement (μέσα σε άγκιστρα)

Παραδείγματα if

```
int income{ 0 }, bonus{ 0 };

cout << "Eisagete to eisodima sas: ";
cin >> income;

if (income < 500)
    cout << "Tha parete epidoma\n";
bonus = 100;

return 0;
```

Compound Statement

```
int income{ 0 }, bonus{ 0 };

cout << "Eisagete to eisodima sas: ";
cin >> income;

if (income < 500) {
    cout << "Tha parete epidoma\n";
    bonus = 100;
}
```

Condition στην if

- Παρατηρήσεις σχετικά με το condition της εντολής if
  - **ΠΡΕΠΕΙ** να είναι λογική τιμή
  - Συνήθως είναι εκφράσεις που χρησιμοποιούν τελεστές σύγκρισης
  - Μπορούν να δημιουργηθούν και ποιο περίπλοκες εκφράσεις με την χρήση των λογικών τελεστών
  - **Προσοχή!** στην προτεραιότητα των τελεστών
  - **Προσοχή!** Μπορεί να γίνει αυτόματη μετατροπή από έναν άλλο τύπο σε λογικό!

Προτεραιότητα τελεστών (μέχρι στιγμής)

Operators	Precedence
!, +, – (unary operators)	first
*, /, %	second
+, –	third
<, <=, >=, >	fourth
==, !=	fifth
&&	sixth
||	seventh
= (assignment operator)	last

Παραδείγματα condition

- Ποια από τα διπλανά conditions είναι συντακτικά σωστά;
- Ποια είναι η σημασία τους;

```
if (income<500)

if ((income<500) && (bonus<100))

if (income)

if (income<500) || (bonus<500)

if (100<bonus<500)
```

Short circuit evaluation

- Στους λογικούς τελεστές (ΚΑΙ , Η) όταν η C++ μπορεί να προβλέψει με ασφάλεια το αποτέλεσμα **ΠΑΡΑΒΛΕΠΕΙ** την αποτίμηση του δεύτερου τελεστή
- Αυτό μπορεί να δημιουργήσει προβλήματα
- Ποια είναι η τιμή της μεταβλητής bonus στον διπλανό κώδικά;

```
int income{ 1000 }, bonus{ 0 };

if ((income < 500) && ++bonus)
    cout << "Value of bonus?";

cout << bonus << endl;
```

if - else

- Η εντολή if μπορεί να έχει και **προαιρετικό** else τμήμα οπού θα περιλαμβάνει εντολές που θα εκτελούνται όταν ΔΕΝ θα ισχύει το condition
- όπως στην if έτσι και στην else εκτελείται μόνο ένα statement. Αν θέλουμε να εκτελούνται παραπάνω από ένα πρέπει να δημιουργήσουμε ένα compound statement (χρήση άγκιστρων)

```
double salary{ 400 };
double bonus{ 0 }, tax{ 0 };

if (salary < 1000)
    bonus = 100;
else
    tax = salary * 0, 1;

cout << "Emp salary:" << salary << " Bo
cout << " Tax:" << tax << endl;
```

Μία εντολή στο else τμήμα

Πολλαπλές επιλογές

Πολλαπλές επιλογές

- Υπάρχουν προβλήματα όπου χρειάζεται η εκτέλεση διαφορετικού κώδικα με βάση παραπάνω από μία condition
  - Π.χ. Ο συντελεστής φορολογίας στα εισοδήματα
- Μπορεί να λυθεί με την χρήση
  - Εμφωλευμένων if
  - Εντολής elseif

Εμφωλευμένες if

- Μπορεί μέσα στις εντολές που υπάρχουν στο if ή στο else statement να υπάρχει και μία καινούργια if
- Σε αυτή την περίπτωση μπορούμε να έχουμε παραπάνω από δύο επιλογές

```
if (income > 1000) {
    tax = 50;
    if (income > 2000)
        tax = 100;
}
else
    cout << "No tax";
```

Κίνδυνος στην εμφωλευμένες if

- Όταν αρχίζουμε και χρησιμοποιούμε εμφωλευμένες if αρχίζει και γίνεται πιο δύσκολα αναγνώσιμος ο κώδικας
- Καλό είναι να χρησιμοποιούμε άγκιστρα και identation για να ανακαλύπτουμε προβλήματα σαν το διπλανό

```
if (fuel < 0.75)
    if (fuel < 0.25)
        cout << "LOW FUEL";
else
    cout << "DONT STOP enough fuel";
```

Η εντολή elseif

- Όταν έχουμε περιπτώσεις πολλαπλών επιλογών που είναι αμοιβαία αποκλειόμενες τότε μπορούμε να χρησιμοποιήσουμε την εντολή else if
- **ΠΡΟΣΟΧΗ** θα εκτελεστεί μόνο μία
- Μπορεί να έχει και προαιρετικό else
- Χρήση άγκιστρων για πολλές εντολές

```
if (income > 2000)
    tax = 200;
else if (income > 15000)
    tax = 150;
else if (income > 10000)
    tax = 100;
else
    tax = 0;
```

Σχεδιασμός elseif

- Όταν σχεδιάζουμε μια else if φροντίζουμε ότι να ξεκινάμε από τις πιο «ειδικές» περιπτώσεις και μετά στις πιο «γενικές»
- Μπορείτε να αναγνωρίσετε το λάθος στο διπλανό κώδικα;

```
if (income > 1000)
    tax = 100;
else if (income > 15000)
    tax = 150;
else if (income > 20000)
    tax = 200;
else
    tax = 0;
```

Εντολή switch

- Όταν θέλουμε να δημιουργήσουμε πολλαπλές επιλογές αλλά βασιζόμενοι στην τιμή μιας παράστασης μπορούμε να χρησιμοποιήσουμε την switch
- Μπορεί να εφαρμοστεί μόνο για integer,char, bool τιμές (σχεδόν)
- Η default είναι προαιρετική
- ΠΡΟΣΟΧΗ στην χρήση της break

```
switch (choice) {
case 1:
    cout << "Epelejes katathesi\n";
    break;
case 2:
    cout << "Epelejes analipsi\n";
    break;
case 3:
    cout << "Epelejes analysi kiniseon\n";
    break;
default:
    cout << "Lathos epilogi\n";
    break;
};
```

Εντολές επανάληψης (loops)

Εντολές επανάληψης

Οι εντολές επανάληψης χωρίζονται σε δύο μεγάλες κατηγορίες

- Σε αυτές που γνωρίζουμε από την αρχή τον αριθμό των επαναλήψεων που θα πραγματοποιηθούν **for**
- Σε αυτές που επαναλαμβάνονται όσο ισχύει μία συνθήκη **while, do while**

Η εντολή while

- Όσο ισχύει το condition επαναλαμβάνονται οι εντολές
- Όπως και στην if
  - Το condition είναι γραμμένο **ΥΠΟΧΡΕΩΤΙΚΑ** μέσα σε παρενθέσεις
  - Αν θέλουμε να επαναλαμβάνονται παραπάνω από μία εντολές χρησιμοποιούμε **compound statement**

```
int value{ 0 };
const int limit = 200;

while (value < limit) {
    cout << "den eftases to orio\n";
    value += 10;
}
```

Παρατηρήσεις στην while

- ΠΡΟΣΟΧΗ στην δημιουργία του condition! Τις περισσότερες φορές στην περιγραφή έχουμε την συνθήκη για το πότε θα σταματήσει η επανάληψη.
  - Εμείς θέλουμε την συνθήκη για **<u>να συνεχίσει η επανάληψη</u>**!
- Μέσα στις εντολές επανάληψης θα πρέπει να υπάρχει μία εντολή που να αλλάζει το αποτέλεσμα του condition. Αλλιώς κινδυνεύουμε με **infinite loop**
- Μπορεί να μην εκτελεστεί **ΚΑΜΙΑ** φορά το while loop

Η εντολή do while

- Η διαφορά με την while είναι το γεγονός ότι η do while θα εκτελέσει τις εντολές της επανάληψης ΤΟΥΛΑΧΙΣΤΟΝ μια φορά
- Αυτό είναι χρήσιμο όταν θέλουμε να διαβάσουμε τιμές από το πληκτρολόγιο
- Προσοχή! Ερωτηματικό στο τέλος

```
int value{ 0 };

do {
    cout << "Dose akeraio (-99 gia telos): ";
    cin>> value;
} while (value != -99);
```

Η εντολή for

- Η εντολή for χρησιμοποιείται όταν γνωρίζουμε τον αριθμό των επαναλήψεων που θα εκτελεστούν
- Δεν κάνει κάτι περισσότερο από την while
- Είναι σε μία πιο συμπαγή μορφή από την while

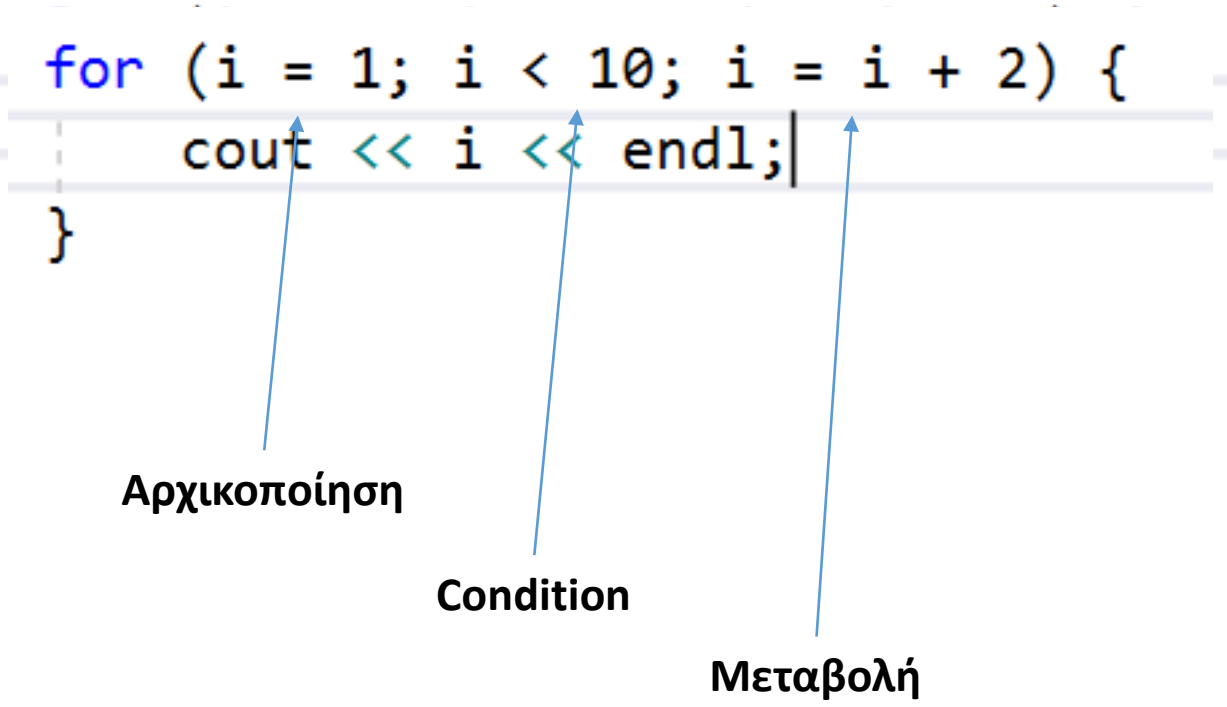
```
for (i = 1; i < 10; i = i + 2) {
    cout << i << endl;
}
```

Λειτουργία της for

Λειτουργία

- Αρχικοποίηση i
- έλεγχος condition
- Σε περίπτωση που είναι αληθές εκτέλεση των εντολών του loop
- Μεταβολή του i

- Ξανά έλεγχος του condition και επανάληψη.
- MEXPI condition false



Παρατηρήσεις την for

- Μπορεί να μην εκτελεστεί καμία φορά
- Μπορεί εκτός από αρχικοποίηση να γίνει και δήλωση του μετρητή. Σε αυτή την περίπτωση έχει εμβέλεια όσο το block της for
- Καθένα από τα τρία μέρη μπορεί να είναι περισσότερο πολύπλοκα
- ΠΡΟΣΟΧΗ στην χρήση ερωτηματικού στην for

```
for (i = pow(3,2); i+6 < 10*pow(3,3); i = i/3 + 2) {
    cout << i << endl;
}
```

Σύγκριση while - for

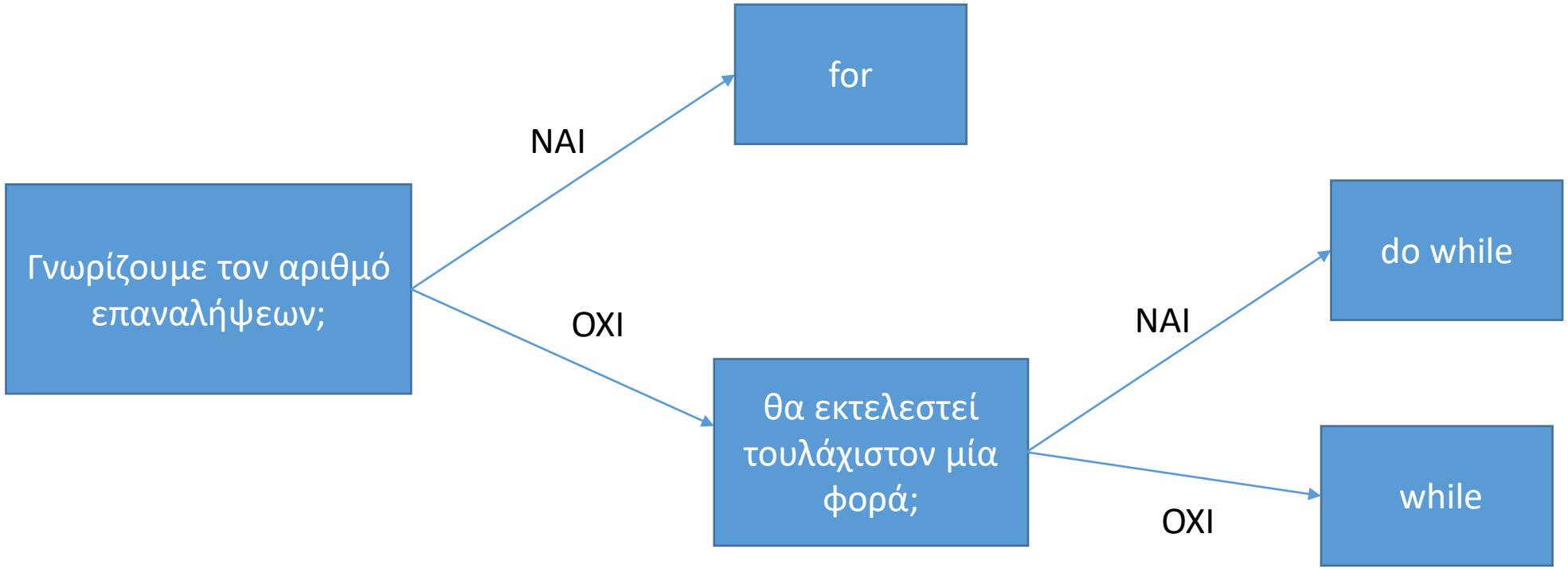
```
for (int i=1; i<10; ++i) {
    cout << i << endl;
}
```

```
int i{ 1 };

while (i<10) {
    cout << i << endl;
    ++i;
}
```

Σχεδιάζοντας Loop

Επιλογή εντολής επανάληψης



Σχεδιασμός loop

- Τα βασικά στοιχεία που πρέπει να αποφασίσουμε κατά την σχεδίαση ενός loop είναι

- Ποιο είναι το condition (το δυσκολότερο)
- Ποιες θα είναι οι εντολές μέσα στο loop
- Ποια θα είναι η αρχικοποίηση του loop
- Πως θα μεταβάλετε σε κάθε loop η εκτέλεση

Αθροίσματα και γινόμενο

- Ένα από τα κλασικά προβλήματα στον προγραμματισμό είναι το άθροισμα/γινόμενο/μέσος όρος αριθμών
- Βλέπουμε ότι αν θέλουμε μπορούμε να μετράμε τον αριθμό επαναλήψεων στην while με την χρήση μιας μεταβλητής

```
int value{ 0 }, count{ 0 };
int sum{ 0 }, prod{ 1 };
double mo{ 0 };

do {
    cout << "Dose num (-99 telos): ";
    cin >> value;
    sum += value;
    prod *= value;
    ++count;
} while (value != -99);

mo = sum / (count - 1);
```

Τρόποι τερματισμού ενός loop

- Βάση μεγέθους
  - Όταν γνωρίζουμε το μέγεθος πριν από την εκτέλεση (for)
- Ερώτηση πριν από κάθε επανάληψη
  - Ρωτά τον χρήστη και παίρνει τιμή πριν από κάθε επανάληψη (do – while)
- Τελειώνει με μία συγκεκριμένη τιμή
  - Ελέγχει την τιμή με την συγκεκριμένη (while)
- Τελειώνει η είσοδος
  - Χρησιμοποιεί ειδικές μεθόδους που ελέγχουν αν έφτασε το τέλος του αρχείου (while)

Ειδικές κατασκευές σε loop

Εντολή if μέσα σε loop

- Πολύ συχνά θέλουμε να επεξεργαστούμε διαφορετικά πολλά δεδομένα
- Αυτό μπορεί να επιτευχθεί με την χρήση μιας εντολής if μέσα στο loop

```
do {
    cout << "Dose bathmo: (-99 telos) ";
    cin >> grade;
    if (grade > 8) {
        cout << "Bravo aristouxos\n";
        ++count;
    }
} while (grade != -99);
```

Εμφωλευμένα loops

- Μπορεί να υπάρχει μια εντολή επανάληψης while/for μέσα στα loop statements μιας άλλης while / for εντολές
- Το αποτέλεσμα είναι να πολλαπλασιάζεται ο συνολικός αριθμός επαναλήψεων
- Παράδειγμα for σε for

```
for (int i = 0; i < 3; ++i) {
    for (int j = 0; j < 2; ++j) {
        cout << "i=" << i << " j=" << j << endl;
    }
}
```

Παράδειγμα for σε while

- Ένα άλλο παράδειγμα είναι να έχουμε άγνωστο αριθμό επαναλήψεων που όμως για κάθε μία τιμή θα έχουμε γνωστό αριθμό επαναλήψεων
- Π.χ. ο χρήστης να δίνει έναν αριθμό και εμείς να εμφανίζουμε την προπαίδεια

```
while(1){
    cout << "dose arithmo: ";
    cin >> base;
    if (base == -99)
        break;
    for (int i = 1; i <= 9; ++i)
        cout << i << "*" << base << '=' << i * base<<endl;
}
```

Οι εντολές break / continue

- Οι εντολές break / continue έχουν σαν σκοπό να μεταβάλλουν δυναμικά τον αριθμό των επαναλήψεων του loop στο οποίο βρίσκονται. Συγκεκριμένα:
- Η break σταματά τις επαναλήψεις
- Η continue σταματά την τρέχουσα επανάληψη

```
for (int i = 1; i <= 10; ++i)
{
    if (i == 6 || i == 9)
    {
        continue;
    }
    cout << i << "\t";
}
```

Συχνά λάθη στα loop

- Infinite loops
  - Λάθος στο condition
  - Λάθος στην μεταβολή της μεταβλητής που συμμετέχει στο condition
- Off-by-one λάθη
  - Είναι τα λάθη όπου οι επαναλήψεις εκτελούνται μία περισσότερη ή λιγότερη φορά
  - Γίνεται σωστά η αρχικοποίηση;
  - Μήπως θέλει >= αντί για == (οι doubles κατά προσέγγιση);
  - Ελέγχει την περίπτωση μηδενικών επαναλήψεων;