# ****Δομές Απόφασης (Selection Statements) στη C++****

Οι δομές απόφασης επιτρέπουν στο πρόγραμμα να **επιλέγει ποιο μπλοκ εντολών θα εκτελέσει** ανάλογα με το αποτέλεσμα μιας συνθήκης.  
Στην C++ υπάρχουν δύο **πραγματικές δομές απόφασης**:

* if (με πιθανό else)
* switch

Οι υπόλοιποι σχηματισμοί, όπως το if – else if – else, θεωρούνται **συνδυασμοί των βασικών αυτών δομών** και όχι ξεχωριστές δομές της γλώσσας.

## ****1. Η δομή if****

Η απλούστερη μορφή απόφασης. Εκτελεί ένα τμήμα κώδικα μόνο αν η συνθήκη είναι αληθής.

if (condition) {

// Εντολές που εκτελούνται όταν η συνθήκη είναι true

}

**Παράδειγμα:**

int x = 5;

if (x > 0) {

cout << "Positive number" << endl;

}

## ****2. Η μορφή if – else****

Η εντολή if μπορεί να συνοδεύεται από ένα else, το οποίο καθορίζει τι θα εκτελεστεί όταν η συνθήκη είναι ψευδής.

if (condition) {

// Εκτελείται όταν η συνθήκη είναι true

} else {

// Εκτελείται όταν η συνθήκη είναι false

}

**Παράδειγμα:**

int x = -3;

if (x >= 0) {

cout << "Non Negative" << endl;

} else {

cout << "Negative" << endl;

}

## ****3. Η τεχνική if – else if – else****

Η C++ **δεν διαθέτει ξεχωριστή εντολή** else if.  
Πρόκειται για **σύνθεση** κατά την οποία το else ακολουθείται από μια δεύτερη εντολή if.  
Η δομή αυτή επιτρέπει αλυσιδωτούς ελέγχους, όπου εξετάζονται διαδοχικά πολλές συνθήκες.

if (grade >= 90)

cout << "Excellent";

else if (grade >= 75)

cout << "Very Good";

else if (grade >= 60)

cout << "Good";

else

cout << "Fail";

Κάθε συνθήκη ελέγχεται **με τη σειρά εμφάνισης** και μόλις μία είναι αληθής, **οι υπόλοιπες παραλείπονται**.  
Είναι λοιπόν **τεχνική οργάνωσης if–else**, όχι νέα γλωσσική δομή.

## ****4. Εμφωλευμένα if (Nested if)****

Η εμφώλευση (δηλαδή ένα if μέσα σε άλλο if) χρησιμοποιείται για **σύνθετες λογικές αποφάσεις**.

int x = 10, y = 20;

if (x > 0) {

if (y > 0) {

cout << "Both numbers are positive" << endl;

}

}

## ****5. Η δομή switch****

Η switch είναι η δεύτερη βασική δομή απόφασης στη C++.  
Επιτρέπει την επιλογή μιας από πολλές εναλλακτικές διαδρομές ανάλογα με την τιμή μιας **έκφρασης ακέραιου ή χαρακτήρα**.

### Γενική μορφή:

switch (expression) {

case value1:

// Εντολές

break;

case value2:

// Εντολές

break;

default:

// Εκτελείται αν δεν ταιριάζει καμία περίπτωση

}

### ****Συμπεριφορά όταν λείπει το break****

Αν παραλείψουμε το break, τότε **η εκτέλεση συνεχίζεται και στις επόμενες περιπτώσεις** (“fall through”) μέχρι να συναντηθεί break ή να τελειώσει η switch.

**Παράδειγμα:**

int x = 2;

switch (x) {

case 1: cout << "One ";

case 2: cout << "Two ";

case 3: cout << "Three ";

}

**Έξοδος:**

Two Three

Αυτό οφείλεται στο ότι η switch δεν σταματά αυτόματα μετά το πρώτο case που ταιριάζει — ο μεταγλωττιστής απλώς **μεταφέρει τον έλεγχο** μέσα στο αντίστοιχο σημείο και συνεχίζει να εκτελεί τις εντολές που ακολουθούν.

Σε κάποιες περιπτώσεις, αυτό το “fall through” είναι **επιθυμητό** και χρησιμοποιείται εσκεμμένα, π.χ.:

switch (day) {

case 1:

cout << “Monday”; break;

case 2:

cout << “Tuesday”; break;

case 3:

cout << “Wednesday”; break;

case 4:

cout << “Thursday”; break;

case 5:

cout << “Friday”; break;

case 6:

case 7:

cout << "Weekend";

break;

default:

cout << "Unkown";

}

## ****6. Ο τριαδικός τελεστής συνθήκης (?:)****

Ο τελεστής ?: είναι **συντόμευση** μιας απλής μορφής if–else.

condition ? expression\_if\_true : expression\_if\_false;

**Παράδειγμα:**

int x = -5;

cout << ((x >= 0) ? "Positive" : "Negative") << endl;

## ****7. Συμπέρασμα****

Οι δομές if και switch αποτελούν τους **θεμελιώδεις μηχανισμούς λήψης αποφάσεων** στη C++.  
Το if προσφέρει **ευελιξία** για σύνθετες συνθήκες, ενώ το switch είναι **καθαρότερο** και πιο **αποδοτικό** όταν πρόκειται για διακριτές σταθερές τιμές.  
Η σωστή χρήση τους αποτελεί βασικό βήμα για τον **έλεγχο της ροής εκτέλεσης** και την ανάπτυξη δομημένων αλγορίθμων.