ΑΣΚΗΣΕΙΣ ΠΡΑΞΗΣ #1

Λίγα για το ολοκληρωμένο περιβάλλον ανάπτυξης (Integrated Development Environment – IDE) ενός προγράμματος που περιλαμβάνει κατ'ελάχιστον έναν συντάκτη και έναν μεταγλωττιστή, όπως ο DEV-C++. Συνήθως, στο περιβάλλον υπάρχουν και εργαλεία αποσφαλμάτωσης (debugging) και παρακολούθησης της εξέλιξης ενός προγράμματος, οργάνωσης σύνθετου προγράμματος που εκτείνεται σε πολλά αρχεία, διαχείρισης βιβλιοθηκών, κ.λπ.

Σε ένα IDE, η διαδικασία περιλαμβάνει τη σύνταξη --> μεταγλώττιση --> σύνδεση με άλλα μεταγλωττισμένα αρχεία και αρχεία βιβιοθηκών και παραγωγή εκτελέσιμου κώδικα.

ΣΥΣΤΑΤΙΚΑ ΕΝΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ: ΜΕΤΑΒΛΗΤΕΣ, ΣΤΑΘΕΡΕΣ, ΠΑΡΑΣΤΑΣΕΙΣ ΚΑΙ ΕΝΤΟΛΕΣ

Μεταβλητή: Θέση μνήμης σε συγκεκριμένη διεύθυνση στην οποία έχει δοθεί ένα όνομα, με μέγεθος ανάλογα με τον **τύπο των δεδομένων** που θα αποθηκεύει (π.χ. 1 byte για χαρακτήρες, 8 bytes για πραγματικούς διπλής ακριβείας, κ.λπ.) και περιεχόμενο που αποτελεί και την τιμή της μεταβλητής. Η πρόσβαση στο περιεχόμενο μιας μεταβλητής γίνεται απλά και μόνο με το όνομά της και ονομάζεται *άμεση προσπέλαση μνήμης*.

Η εντολή που δίνει μια τιμή σε μια μεταβλητή (δηλαδή νέο περιεχόμενο στη θέση μνήμης της μεταβλητής) ονομάζεται εντολή εκχώρησης. Η εκχώρηση μιας αρχικής τιμής σε κάποια μεταβλητή ονομάζεται αρχικοποίηση της μεταβλητής.

Οι τύποι δεδομένων και το εύρος τιμών τους ευρίσκονται στο αρχείο κεφαλίδας <limits.h>.

Παραδείγματα δήλωσης μεταβλητών:

```
int  x;

float  a, b;

char  ch = 'A';

int  v1 = 100, v2, v3 = v1 + 100, v4 = v1 + v3;
```

Οι δεσμευμένες λέξεις signed και unsigned χρησιμοποιούνται για να καθορίσουμε αν οι τιμές που θα αποθηκεύονται στις μεταβλητές είναι προσημασμένες ή μη προσημασμένες.

Το default είναι signed εκτός ίσως (σε κάποιους μεταγλωττιστές) για τον τύπο char που μπορεί να είναι απροσδιόριστος. Σ'αυτήν την περίπτωση, αν θέλουμε να έχουμε κάποια προσημασμένη ακέραια μεταβλητή μεγέθους 1 byte (στο διάστημα -128 έως +127) τότε θα την δηλώναμε ως εξής:

```
signed char  a;
```

Ένας ακέραιος μήκους 2 bytes μπορεί να δηλωθεί ως short int ή απλά short. Αντίστοιχα για έναν ακέραιο των 4 bytes ο τύπος είναι long int ή απλά long. Ο τύπος int είναι συνήθως των 4 bytes (μόνο σε κάποια πολύ παλιά συστήματα είναι των 2 bytes). Ισχύουν οι ακόλουθες ανισότητες ως προς τα μεγέθη των ακέραιων τύπων δεδομένων:

sizeof (char) < sizeof (short) ≤ sizeof (int) ≤ sizeof (long) ≤ sizeof (long long)

Αντίστοιχα:

sizeof (float) ≤ sizeof (double) ≤ sizeof (long double)

Ακρίβεια τύπου float = 6 δεκαδικά ψηφία. Ακρίβεια double = 15 δεκαδικά ψηφία. Αν δεν μας ενδιαφέρει τόσο μεγάλη ακρίβεια, προτιμάμε τον τύπο float για λόγους ταχύτητας εκτέλεσης των πράξεων (κυρίως) παρά για εξοικονόμηση μνήμης.

Σταθερές

Ακέραιες σταθερές: δεκαδικές (default), hex (αρχίζουν με 0x ή 0X, π.χ. 0x10fe), octal (αρχίζουν με 0, π.χ. 0172).

Παράδειγμα: $107_{10} = 01101011_2 = 153_8 = 6B_{16}$ ή 107 = 0153 = 0x6B

```
int  a = 0xFF26, b = 0126, c = 126;

signed char  c1 = 150, c2 = 260 ;    /* c1, c2 εκτός ορίων:  -128 <= c <= +127 */
```

$150 = 10010110_2$ ==> αρνητικό πρόσημο ==> 2's complement ==> c1 = - (256 – 150) = -106

$260 = 1|00000100_2$ ==> χάνεται το πιο σημαντικό bit ==> c2 = $00000010_2 = 4$

Πραγματικές σταθερές: εξ' ορισμού οι πραγματικές σταθερές θεωρούνται διπλής ακριβείας (double). Έτσι, αν η σταθερά αυτή εκχωρείται σε μεταβλητή τύπου float θα πρέπει να γίνει μετάπτωση σε αναπαράσταση μικρότερης ακρίβειας (από 8 σε 4 bytes). Αυτή η μετάπτωση μπορεί να γίνει αυτόματα κατά την εκχώρηση σε μεταβλητή μικρότερης ακριβείας όπως στο παράδειγμα

```
float pi = 3.14159;
```

ή μπορούμε να υποχρεώσουμε την σταθερά να είναι απλής ακριβείας με την κατάληξη f ή F

```
float pi = 3.14159f;
```

Αν θέλουμε σταθερά long double τότε η κατάληξη πρέπει να είναι l ή L:

```
long double z = 3.14159265358979323846264 3L
```

Πριν τον υπολογισμό μιας αριθμητικής παράστασης, οι τελεστέοι πρέπει να μετατραπούν (αν δεν είναι) στον ίδιο τύπο δεδομένων. Αυτό μπορεί να γίνει είτε άμεσα με προσωρινή αλλαγή τύπου από τον προγραμματιστή (type casting) είτε έμμεσα, δηλαδή αυτόματα (από τον μεταγλωττιστή), όπου ο γενικός κανόνας είναι να προβιβάζεται αυτόματα ο τύπος του τελεστέου με τον λιγότερο ευρύ τύπο στον τύπο του τελεστέου με τον ευρύτερο τύπο:

int --> float --> double --> long double

char --> short --> int --> unsigned int --> long --> unsigned long (αν long και int έχουν το ίδιο εύρος, τότε και οι δύο τελεστέοι μετατρέπονται σε unsigned long)

Προσοχή στον συνδυασμό απρόσημων και προσημασμένων τύπων. Παράδειγμα:

```
int a = -10;
unsigned int b = 100;
if (a < b)
  printf("Yes\n");
else
  printf("No\n");
```

Type casting examples:

float a, b, c = 2.34;

a = (int) c; /* a = 2 */

b = (int) (c + 1.5); /* b = 3 */

b = (int) c + 1.5; /* b = 3.5 */

Για ορισμό σταθερών μπορούμε να χρησιμοποιήσουμε το προσδιοριστικό τύπου **const** μπροστά από τον τύπο δεδομένων και το όνομα της μεταβλητής. Στην ουσία με το const ορίζουμε μεταβλητές που δεν επιτρέπεται να αλλάξουν τιμή κατά την εκτέλεση του προγράμματος.

const int a = 5;

const float pi = 3.14159;

Το προσδιοριστικό τύπου **volatile** ενημερώνει τον μεταγλωττιστή ότι η τιμή της μεταβλητής μπορεί να αλλάξει αν και αυτό δεν είναι φανερό στον μεταγλωττιστή και τον αποτρέπει από σχετική βελτιστοποίηση κώδικα που μπορεί να απαλείψει έλεγχο της τιμής της μεταβλητής

Π.χ.
```
int  flag = 0;
while (flag != 100)
{
    ...
}
```

βελτιστοποιείται σε

```
int  flag = 0;

while (1)
{
```

Η δήλωση της μεταβλητής ως

```
volatile int flag = 0;
```

αποτρέπει την παραπάνω βελτιστοποίηση.