ΕΒΔΟΜΗ ΔΙΑΛΕΞΗ

Έννοια συνάρτησης

- Ένα σύνολο εντολών που επιτελούν κάτι συγκεκριμένο, έχουν σημαντική σύνδεση μεταξύ τους και μικρή εξάρτηση από άλλες, μπορεί να απομονωθεί σε αυτόνομη συνάρτηση.
- Το σύνολο θα έχει συγκεκριμένο όνομα και θα παραμετροποιείται από καμία, μία ή περισσότερες ποσότητες, τα ορίσματα της συνάρτησης. Υπάρχει η δυνατότητα να παραμετροποιείται και με τύπους ποσοτήτων.
- Το σύνολο θα «καλείται» στον υπόλοιπο κώδικα όπου και όσες φορές χρειάζεται με το όνομά του, αφού προσδιορίσουμε κατάλληλες τιμές για τα ορίσματα της συνάρτησης. Στο σημείο της κλήσης υπάρχει η δυνατότητα να «επιστρέφεται» ως αποτέλεσμα μία ποσότητα ή να τροποποιούνται τα ορίσματα.

Ορισμός συνάρτησης (1/2)

Ο ορισμός μιας συνάρτησης έχει την ακόλουθη γενική μορφή:

τύπος_επιστρεφόμενης_ποσότητας
όνομα(τύπος_ορίσματος_Α όρισμαΑ,
τύπος_ορίσματος_Β όρισμαΒ, ...)
{
// κώδικας
}

Το τμήμα μεταξύ των {}, συμπεριλαμβανομένων αυτών, αποτελεί το σώμα της συνάρτησης και οι εντολές που περιλαμβάνει είναι αυτές που θα εκτελεστούν όταν κληθεί η συνάρτηση.

Ορισμός συνάρτησης (2/2)

Παρατηρήσεις

- Ο «τύπος_επιστρεφόμενης_ποσότητας» μπορεί να είναι **`void`**· υποδηλώνεται έτσι ότι δεν επιστρέφεται τιμή.
- Η λίστα ορισμάτων μπορεί να είναι κενή ή, ισοδύναμα, να περιέχει τη λέξη **`void`**. Τα ορίσματα, αν υπάρχουν, δεν μπορούν να επαναοριστούν στο σώμα της συνάρτησης και η εμβέλειά τους εκτείνεται ως το καταληκτικό '}' του σώματος.
- Αν τυχόν υπάρχει κάποιο όρισμα της συνάρτησης που δεν χρησιμοποιείται στο σώμα της, μπορούμε να παραλείψουμε το όνομά του (αλλά όχι τον τύπο του) στη λίστα ορισμάτων.
- Ο ορισμός μιας συνάρτησης γράφεται έξω από οποιαδήποτε συνάρτηση.

Επιστροφή τιμής από συνάρτηση

- Η επιστροφή τιμής, ως αποτέλεσμα, από τη συνάρτηση και η διακοπή της εκτέλεσής της γίνεται με την εντολή

  return *τιμή;*

  που πρέπει να εμφανίζεται μία ή περισσότερες φορές στο σώμα της συνάρτησης.
- Η συνάρτηση `main()` αποτελεί εξαίρεση: το **`return`** μπορεί να παραληφθεί οπότε θεωρείται ως τελευταία εκτελέσιμη γραμμή η εντολή **`return`** `0;`.
- Η «τιμή» που επιστρέφεται μπορεί να είναι μια μεταβλητή ή σταθερή ποσότητα ή να υπολογίζεται από έκφραση. Ο τύπος της πρέπει να είναι ο τύπος της ποσότητας που επιστρέφει η συνάρτηση ή να μπορεί να μετατραπεί σε αυτόν.
- Μια συνάρτηση που δεν επιστρέφει τιμή («επιστρέφει» **`void`**), μπορεί, χωρίς να είναι απαραίτητο, να περιλαμβάνει εντολές **`return`**`;` (χωρίς τιμή).

Δήλωση συνάρτησης

- Μια συνάρτηση για να κληθεί σε κάποιο αρχείο πρέπει προηγουμένως να έχει οριστεί ή να έχει δηλωθεί στο αρχείο.
  Ο compiler πρέπει να γνωρίζει
  - το όνομά της,
  - τα ορίσματα (τύπο και πλήθος τους) και
  - τον τύπο επιστρεφόμενης ποσότητας

  ώστε να ελέγξει αν γίνεται σωστά η κλήση.
- Η δήλωση είναι ακριβώς η ίδια με τον ορισμό αλλά το σώμα της συνάρτησης έχει αντικατασταθεί από το ';' :

  τύπος_επιστρεφόμενης_ποσότητας
  όνομα(τύπος_ορίσματος_Α όρισμαΑ,
  τύπος_ορίσματος_Β όρισμαΒ, ...);

  Στη δήλωση της συνάρτησης τα ονόματα των μεταβλητών μπορούν να παραληφθούν.

Κλήση συνάρτησης

- Η κλήση μιας συνάρτησης γίνεται παραθέτοντας το όνομά της, ακολουθούμενο σε παρενθέσεις από ποσότητες κατάλληλου τύπου και πλήθους ώστε να αντιστοιχούν στα ορίσματά της (ή να μπορούν να μετατραπούν σε αυτά).
- Στην περίπτωση που έχουμε προκαθορισμένες τιμές για κάποια ορίσματα (από το τέλος), η κλήση μπορεί να γίνει με λιγότερες τιμές από τα ορίσματα.
- Δεν είναι απαραίτητο να χρησιμοποιήσουμε το (τυχόν) αποτέλεσμα της συνάρτησης. Μπορεί να θέλουμε απλά να εκτελεστεί η συνάρτηση και να προκαλέσει "παρενέργειες".

Παράδειγμα συνάρτησης

Η μαθηματική συνάρτηση $f(x) = x^2 + 5x + \sqrt{x}$ γράφεται σε κώδικα ως εξής

Ορισμός

```
double f(double x)
{
   return x*x + 5*x + std::sqrt(x);
}
```

Δήλωση

```
double f(double x);
```

Κλήση

```
auto y = f(3.2);
f(4.5);   // επιτρεπτό
```

Δήλωση ορισμάτων

- Οι τιμές των ποσοτήτων που δίνονται κατά την κλήση στη συνάρτηση χρησιμοποιούνται ως αρχικές τιμές σε "εσωτερική" δήλωση των ορισμάτων. Οι νέες μεταβλητές ή οι αναφορές έχουν εμβέλεια το σώμα της συνάρτησης.
- Αν ένα όρισμα δεν είναι αναφορά, η ποσότητα που του δίνεται κατά την κλήση αποτελεί αρχική τιμή σε νέα μεταβλητή, δηλαδή αντιγράφεται στο όρισμα. Οι αλλαγές στο σώμα δεν μεταφέρονται.
- Αν ένα όρισμα είναι αναφορά, η ποσότητα που του δίνεται κατά την κλήση ταυτίζεται με το όρισμα. Κάθε αλλαγή στο όρισμα μεταφέρεται και στην αρχική μεταβλητή.
- Αν θέλουμε να αποφύγουμε την αντιγραφή τιμής αλλά να μην δίνουμε δυνατότητα τροποποίησης, η δήλωση του ορίσματος πρέπει να είναι σταθερή αναφορά (`const &`).

Δήλωση ορισμάτων

Παράδειγμα Α'

```
#include <iostream>

void add3(int x, int & y)
{
   x += 3;
   y += 3;
}

int main()
{
   int a{1}, b{1};
   add3(a, b);
   std::cout << a << ' ' << b << '\n'; // a -> 1, b -> 4
}
```

Δήλωση ορισμάτων

Παράδειγμα Β'

```
#include <fstream>
#include <vector>
#include <string>

void f(std::vector<int> const & a, std::string const & fname)
{
   std::ofstream out{fname};
   for (auto const & x : a) {
       out << x << '\n';
   }
}

int main()
{
   std::vector<int> a(1000, 23);
   f(a, "data.txt");
}
```

Προκαθορισμένες τιμές ορισμάτων

Σε μια συνάρτηση μπορεί να δηλωθεί ότι ένα ή περισσότερα *από το τέλος*, ή και όλα τα ορίσματά της, παίρνουν προκαθορισμένες τιμές:

```
void f(double a, double b = 5.0, double c = 3.0);
```

Η κλήση της `f()` μετά από τέτοια δήλωση μπορεί να γίνει

- με τρία ορίσματα:

  ```
  f(2.1, 3.2, 4.5);
  ```

- με δύο ορίσματα (που αντιστοιχούν στα a,b):

  ```
  f(2.1, 3.2);
  ```

  Το c παίρνει την προκαθορισμένη τιμή (3.0).

- με ένα όρισμα (που αντιστοιχεί στο a):

  ```
  f(2.1);
  ```

  Τα b, c παίρνουν τις προκαθορισμένες τιμές τους (5.0, 3.0).

Οργάνωση κώδικα σε μεγάλα προγράμματα

- Γράφουμε πολλές μικρές συναρτήσεις (~ 25 γραμμών)
- Η δήλωση συνάρτησης προτείνεται να περιλαμβάνεται σε αρχείο header με κατάληξη .hpp, το οποίο συμπεριλαμβάνεται σε κάθε αρχείο στο οποίο θα κληθεί η συνάρτηση καθώς και στο αρχείο που έχει τον ορισμό της συνάρτησης.
- Ο ορισμός συνάρτησης προτείνεται να περιλαμβάνεται σε αρχείο με κατάληξη .cpp, μόνος του ή μαζί με τις βοηθητικές του συναρτήσεις, που δε χρησιμοποιούνται αλλού. Το συγκεκριμένο αρχείο κάνει `#include` το αρχείο header που περιλαμβάνει τη δήλωση της συνάρτησης (και των τυχόν βοηθητικών συναρτήσεων).
- Κατά τη μεταγλώττιση, κάθε αρχείο .cpp μεταγλωττίζεται χωριστά, παράγοντας αρχείο object. Στη φάση του linking, συνδέουμε όλα τα αρχεία object ώστε να παραχθεί το εκτελέσιμο αρχείο.

Αναδρομική (recursive) συνάρτηση

Στη C++ επιτρέπεται σε μια συνάρτηση να καλεί τον εαυτό της. Μια συνάρτηση που καλεί τον εαυτό της απλοποιεί πολύ οποιοδήποτε πρόβλημα, αρκεί ο αλγόριθμος επίλυσής του να μπορεί να γραφεί ώστε:

- ο υπολογισμός του αποτελέσματος να χρειάζεται την εφαρμογή του ίδιου αλγόριθμου αλλά σε διαφορετικές «τιμές» για τα δεδομένα εισόδου από αυτές που δέχτηκε αρχικά,
- ο αλγόριθμος να μπορεί να υπολογίσει το αποτέλεσμα για ένα συγκεκριμένο σύνολο «τιμών» με άλλο τρόπο και όχι με εφαρμογή του εαυτού του. Το συγκεκριμένο σύνολο πρέπει να μπορεί να το «φτάσει» σε κάποια από τις διαδοχικές εφαρμογές του εαυτού του.

Αναδρομική (recursive) συνάρτηση

Παράδειγμα

Το παραγοντικό ενός ακεραίου (n!) ορίζεται αναδρομικά:

$$n! = \begin{cases} 1 \times 2 \times \cdots \times (n-1) \times n = (n-1)! \times n\ , & n > 0\ , \\ 1\ , & n = 0\ . \end{cases}$$

Ο υπολογισμός του n! απαιτεί τον υπολογισμό του παραγοντικού ενός άλλου ακεραίου (του n−1). Επιπλέον, για n= 0 ο υπολογισμός γίνεται με άλλο τρόπο (απευθείας) και όχι με υπολογισμό άλλου παραγοντικού.

Κώδικας

```
std::size_t factorial(std::size_t n)
{
  return (n > 0) ?  n * factorial(n-1) : 1;
}
```

Εκτέλεση

- Το `factorial(3)` είναι το `3*factorial(2)`.
- Το `factorial(2)` είναι το `2*factorial(1)`.
- Το `factorial(1)` είναι το `1*factorial(0)`.
- Το `factorial(0)` είναι το `1`.

Συνάρτηση ως όρισμα

- Μπορούμε να έχουμε ως όρισμα συνάρτησης μια άλλη συνάρτηση ώστε να μπορούμε να εφαρμόσουμε ένα αλγόριθμο σε πολλές συναρτήσεις χωρίς να δεσμευόμαστε από κάποιο συγκεκριμένο όνομα αλλά μόνο από τον (κοινό) τύπο τους.
- Ο μηχανισμός για τη δήλωση του ορίσματος παρέχεται από το header `<functional>`. Δηλώνουμε ότι το όρισμα είναι `std::function<>` με παράμετρο εντός των `<>` τον τύπο της συνάρτησης που θέλουμε να καλέσουμε. Σε αυτόν παραλείπουμε τα ονόματα της συνάρτησης και των ορισμάτων της.
- Κατά την κλήση, περνάμε ως όρισμα μόνο το όνομα κάποιας συνάρτησης, αρκεί αυτή να έχει σωστό τύπο.

Συνάρτηση ως όρισμα

Παράδειγμα

```
#include <functional>
#include <iostream>

double g1(double x) { return x*x;     }
double g2(double x) { return 1/(x+5); }

double aver(double a, double b,
            std::function<double (double)> f)
{   return (f(a)+f(b))/2.0;   }

int main()
{
    std::cout << aver(2.0, 4.0, g1) <<'\n';
    std::cout << aver(2.0, 4.0, g2) <<'\n';
}
```

Overloading

- Επιτρέπεται να έχουμε στο ίδιο πρόγραμμα, περισσότερες από μία συναρτήσεις με το ίδιο όνομα.
- Πρέπει να διαφέρουν στο πλήθος ή/και στους τύπους των ορισμάτων. Ο τύπος της επιστρεφόμενης ποσότητας δεν λαμβάνεται υπόψη.
- Κατά την κλήση, ο μεταγλωττιστής κρίνει από το πλήθος ή/και τους τύπους των ορισμάτων (με τις επιτρεπτές μετατροπές) ποια συνάρτηση από όλες πρέπει να καλέσει.