Η δεσμευμένη λέξη const

#5

Τμήμα Πληροφορικής και Τηλεπικοινωνιών
Πανεπιστήμιο Ιωαννίνων (Άρτα)
Γκόγκος Χρήστος

const

- Γενικά, η δεσμευμένη λέξη const εφαρμόζεται σε αναγνωριστικά από τον προγραμματιστή προκειμένου να δηλώσει την πρόθεσή του ότι το αναγνωριστικό δεν θα πρέπει να χρησιμοποιείται για να αλλάξει δεδομένα στα οποία αναφέρεται το αναγνωριστικό.
- Ο μεταγλωττιστής επιβάλλει την πρόθεση του προγραμματιστή, για παράδειγμα (t1.cpp):

```
int main() {
        const int x = 5;
        x = 2;
}
```

```
g++ t1.cpp
t1.cpp:4:5: error: cannot assign to variable 'x' with const-qualified type 'const int'
  x = 2;
  ~ ^
t1.cpp:3:13: note: variable 'x' declared const here
  const int x = 5;
  ~~~~~~~~~~^~~~~
1 error generated.
```

- Στο παραπάνω παράδειγμα, το x είναι ένα αναγνωριστικό που αναφέρεται σε ακέραια δεδομένα στη μνήμη και ο μεταγλωττιστής **επιβάλλει** ότι το x δεν θα πρέπει να χρησιμοποιείται με οποιοδήποτε τρόπο που θα οδηγούσε σε αλλαγή των δεδομένων.

https://github.com/chgogos/oop/blob/master/various/COP3330/lect5/t1.cpp

const

- Σε ένα άλλο παράδειγμα (t2.cpp):

```
int main() {
        const int x = 5;
        int *x_ptr;
        x_ptr = & x;
}
```

```
g++ t2.cpp
t2.cpp:6:11: error: assigning to 'int *' from incompatible type 'const int *'
  x_ptr = & x;
        ^~~
1 error generated.
```

- Στο παραπάνω παράδειγμα, ο μεταγλωττιστής δεν επιτρέπει στο δείκτη x_ptr να δείξει προς τη διεύθυνση του x (που είναι τύπου const int). Αυτό συμβαίνει διότι ο x_ptr θα μπορούσε να χρησιμοποιηθεί για να αλλάξει τα δεδομένα που είναι αποθηκευμένα στο x.

https://github.com/chgogos/oop/blob/master/various/COP3330/lect5/t2.cpp

Επισκόπηση: const παράμετροι

- Στη C++ υπάρχουν δύο τρόποι με τους οποίους μπορούν να περάσουν ορίσματα στις συναρτήσεις:
  - **Πέρασμα με τιμή** (call by value): Ένα αντίγραφο του ορίσματος δημιουργείται και η συνάρτηση επενεργεί σε αυτό το αντίγραφο.
  - **Πέρασμα με αναφορά** (call by reference): Δεν δημιουργείται αντίγραφο, η παράμετρος της συνάρτησης είναι ένα αναγνωριστικό που αναφέρεται στα ίδια δεδομένα με το όρισμα (η παράμετρος λειτουργεί ως ένα **ψευδώνυμο** για το όρισμα).
- Η μέθοδος που χρησιμοποιείται για το πέρασμα των ορισμάτων υποδηλώνεται από τις παραμέτρους της συνάρτησης:

```
void foo(int x);  // το x περνά με τιμή
void foo(int &x); // το x περνά με αναφορά
```

- Ποια είναι η κύρια διαφορά ανάμεσα στους δύο αυτούς μηχανισμούς περάσματος παραμέτρων (δείτε το t3.cpp);

https://github.com/chgogos/oop/blob/master/various/COP3330/lect5/t3.cpp

Επισκόπηση: const παράμετροι

- Ο προγραμματιστής μπορεί να προσθέτει τη δεσμευμένη λέξη const και στους δύο μηχανισμούς περάσματος παραμέτρων (t4.cpp):

```
void foo(const int x);  /* το x είναι μια παράμετρος εισόδου
(μόνο για ανάγνωση), δεν μπορεί να τροποποιηθεί εντός της foo */
void foo(const int &x); /* το x είναι μια παράμετρος εισόδου
(μόνο για ανάγνωση), δεν μπορεί να τροποποιηθεί εντός της foo */
```

- Ποια είναι η διαφορά ανάμεσα στους δύο αυτούς μηχανισμούς περάσματος παραμέτρων;
  - Το πέρασμα με τιμή πρέπει να κατασκευάσει ένα αντίγραφο.
  - Το πέρασμα με αναφορά δεν κατασκευάζει αντίγραφο (απλά περνά την αναφορά της πραγματικής παραμέτρου στο υποπρόγραμμα)
    - Η αναφορά μιας μεταβλητής είναι απλά ένας δείκτης προς τη μεταβλητή (8 bytes)
    - Το πέρασμα με const αναφορά είναι πολύ χρήσιμο όταν η παράμετρος είναι μια μεγάλη δομή δεδομένων έτσι ώστε να μειώσει την επιβάρυνση που θα δημιουργούταν από την κατασκευή ενός αντιγράφου.
- Δείτε το t4.cpp.

https://github.com/chgogos/oop/blob/master/various/COP3330/lect5/t4.cpp

Πέρασμα αντικειμένων με const αναφορά

- Τα αντικείμενα μπορούν επίσης να περάσουν με const αναφορά έτσι ώστε να αποφευχθεί η επιβάρυνση αντιγραφής:

```
friend Fraction Add(const Fraction& f1, const Fraction& f2);
```

- Όπως και με τους άλλους τύπους δεδομένων, ο μεταγλωττιστής διασφαλίζει ότι ένα αντικείμενο που περνά με const αναφορά δεν θα χρησιμοποιηθεί με τέτοιο τρόπο που θα μπορούσε να αλλάξει τα μέλη δεδομένα του.

const συνάρτηση μέλος

- Οποιαδήποτε κλήση σε μια συνάρτηση μέλος διαθέτει το «καλών αντικείμενο»

```
Fraction f1;   // ένα αντικείμενο Fraction
f1.Evaluate(); // το f1 είναι το καλών αντικείμενο
```

- Καθώς η συνάρτηση μέλος έχει πρόσβαση στα δεδομένα του καλούντος αντικειμένου, μπορεί να θέλουμε να διασφαλίσουμε ότι το καλών αντικείμενο δεν πρόκειται ποτέ να αλλάξει από τη συνάρτηση μέλος.
- Τέτοιες συναρτήσεις ονομάζονται **const συναρτήσεις μέλη** και υποδηλώνονται από τη χρήση της δεσμευμένης λέξης const μετά τη δήλωση της συνάρτησης καθώς **και** στον ορισμό της υλοποίησης της συνάρτησης.
- Δείτε το t5.cpp.

https://github.com/chgogos/oop/blob/master/various/COP3330/lect5/t5.cpp

const αντικείμενα

- const μεταβλητές είναι εκείνες οι μεταβλητές που έχουν μόνο μια τιμή (αυτή με την οποία αρχικοποιήθηκαν).

```
const int SIZE = 10;
const double PI = 3.1415;
```

- Τα αντικείμενα μπορούν να δηλωθούν με παρόμοιο τρόπο ως const. Ο κατασκευαστής αρχικοποιεί τα const αντικείμενα, αλλά μετά από αυτό, τα μέλη δεδομένα του αντικειμένου δεν θα μπορούν να αλλάξουν.

```
const Fraction ZERO; // το κλάσμα ορίζεται σε 0/1 και δεν αλλάζει
const Fraction FIXED(3,4);// το κλάσμα ορίζεται σε 3/4 και δεν αλλάζει
```

- Για να διασφαλιστεί ότι ένα αντικείμενο δεν μπορεί να αλλάξει, ο μεταγλωττιστής επιβάλλει ότι ένα const αντικείμενο μπορεί να καλεί μόνο const συναρτήσεις μέλη.
- Δείτε το παράδειγμα const_fraction.cpp.

https://github.com/chgogos/oop/tree/master/various/COP3330/lect5/const_fraction

const μέλη δεδομένων

- Τα μέλη δεδομένων μιας κλάσης μπορούν επίσης να δηλωθούν ως const, αλλά υπάρχουν ορισμένοι συντακτικοί κανόνες που περιπλέκουν την κατάσταση.
- Γνωρίζουμε ότι όταν μια μεταβλητή δηλώνεται με το const σε ένα μπλοκ κώδικα, θα πρέπει να αρχικοποιείται στην ίδια γραμμή:

```
const int SIZE = 10;
```

- Ωστόσο, δεν είναι συντακτικά σωστό να αρχικοποιούμε μεταβλητές μέλη δεδομένων στις γραμμές κώδικα στις οποίες δηλώνονται στην κλάση (παρατήρηση: από την έκδοση C++11 και μετά αυτό επιτρέπεται).
- Αλλά μια const δήλωση μεταβλητής δεν μπορεί «να σπάσει» σε ένα κανονικό τμήμα κώδικα σε 2 γραμμές (δήλωση και ανάθεση).
- Επίσης, η απόπειρα να αναθέσουμε τιμή στην const μεταβλητή μέλος μέσα σε έναν κατασκευαστή δεν θα λειτουργήσει.

Λίστα αρχικοποίησης

- Μπορούμε να χρησιμοποιήσουμε μια ειδική περιοχή του κατασκευαστή που ονομάζεται **λίστα αρχικοποίησης (initialization list)** έτσι ώστε να υπερκεράσουμε το πρόβλημα της αρχικοποίησης const μελών αντικειμένων.
- Οι λίστες αρχικοποίησης έχουν την ακόλουθη μορφή:

```
classname::classname(p1, p2): const_member_var1(p1), member_var2(p2)
{
        // κώδικας
}
```

- Η παραπάνω λίστα αρχικοποίησης θα θέσει το const_member_var1 και το member_var2 στις τιμές που περνάνε στον κατασκευαστή ως p1 και p2 αντίστοιχα.

Ερωτήσεις σύνοψης

- Περιγράψτε το πέρασμα με τιμή και το πέρασμα με αναφορά.
- Γιατί χρησιμοποιούμε το πέρασμα με αναφορά;
- Σε τι αναφέρεται ο όρος «καλών αντικείμενο»;
- Τι είναι μια const συνάρτηση μέλος;
- Τι είναι ένα const αντικείμενο;
  - Πώς το αρχικοποιούμε;
- Τι είναι const μέλος δεδομένων;
  - Πώς το αρχικοποιούμε;
- Μέσω ποιου μηχανισμού το const επιβάλλεται;

Αναφορές

- http://www.cs.fsu.edu/~xyuan/cop3330/