ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΥΠΟΛΟΓΙΣΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Επεξήγηση γλώσσας C++

Καραλής Γιώργος

Εισηγητής: Δρ Νικόλαος Ζάχαρης Καθηγητής

ΑΘΗΝΑ
Ιούνιος 2018

Επεξήγηση γλώσσας C++

ΚαραλήςΓιώργος

Επεξήγηση γλώσσας C++

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ
Καραλής Γιώργος
Α.Μ. 43614

Εισηγητής:

Εισηγητής: Δρ Νικόλαος Ζάχαρης Καθηγητής

Εξεταστική Επιτροπή:

Ημερομηνία εξέτασης 15-6-2018

Επεξήγηση γλώσσας C++

ΚαραλήςΓιώργος

Επεξήγηση γλώσσας C++

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

ΚαραλήςΓιώργος

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα πτυχιακή εργασία ολοκληρώθηκε μετά από επίμονες προσπάθειες, σε ένα ενδιαφέρον γνωστικό αντικείμενο, όπως αυτό της επεξεργασίας κειμένου. Την προσπάθειά μου αυτή υποστήριξε ο επιβλέπων καθηγητής μου, τον οποίο θα ήθελα να ευχαριστήσω.

Ακόμα θα ήθελα να ευχαριστήσω την οικογένεια μου που θα ήθελε να τελειώσω τις σπουδές μου σε λιγότερο από οκτώ χρόνια.

Επεξήγηση γλώσσας C++

ΠΕΡΙΕΧΟΜΕΝΑ.

ΚαραλήςΓιώργος

Επεξήγηση γλώσσας C++

1), Αρχεία ήχου.

Αρχεία ήχου "mp3" και wav.
Το mp3 όπως και το wav κατάσονται στα αρχεία ήχου. Εκτώς από αυτά τα δύο υπάρχουν και άλλα όπως το mp1,mp2,mp3,mp4,wav,MIDI και AIF.

1=Το MPEG-1 Audio Layer I, συνήθως συντομογραφούμενο σε MP1, είναι ένα από τα τρία μορφότυπα ήχου που περιλαμβάνονται στο πρότυπο MPEG-1. Πρόκειται για μια σκόπιμα απλοποιημένη έκδοση του MPEG-1 Audio Layer II, που δημιουργήθηκε για εφαρμογές όπου η χαμηλότερη αποδοτικότητα συμπίεσης θα μπορούσε να είναι ανεκτή(να φτασει ως τα ορια) σε αντάλλαγμα για έναν λιγότερο περίπλοκο αλγόριθμο που θα μπορούσε να εκτελεστεί με απλούστερες απαιτήσεις υλικού. Ενώ υποστηρίζεται από τους περισσότερους φορείς αναπαραγωγής πολυμέσων, ο κωδικοποιητής θεωρείται σε μεγάλο βαθμό απαρχαιωμένος(λανθασμενος) και αντικαθίσταται από MP2 ή MP3.
Για αρχεία που περιέχουν μόνο ήχο MP1, χρησιμοποιείται η κατάληξη αρχείου .mp1.
Το στρώμα MPEG-1 χρησιμοποιήθηκε επίσης από τη μορφή ψηφιακής κασέτας, με τη μορφή κωδικοποιητή συμπίεσης ήχου PASC. Λόγω της ανάγκης για μια σταθερή ροή καρέ ανά δευτερόλεπτο σε ένα μέσο που βασίζεται σε ταινία, το PASC χρησιμοποιεί το σπάνια χρησιμοποιούμενο (και υπο-τεκμηριωμένο) pad padding στην κεφαλίδα MPEG.

2=Μορφή βίντεο MPEG-2. Μορφή συμπίεσης ήχου MPEG-1 Audio Layer II και μορφή αρχείου .mp2. Θεωρία διαταραχών Moller-Plesset της δεύτερης τάξης.

3=Το MPEG-1 Audio Layer 3 (3ο Επίπεδο Ήχου γνωστό και ως MP3 , είναι ένα δημοφιλές πρότυπο ψηφιακής κωδικοποίησης ήχου, το οποίο βασίζεται στην πιό αποτελεσματική συμπίεση αρχείων μέσω ενός αλγορίθμου σχεδιασμένου να μειώνει δραστικά το πλήθος των ψηφιακών δεδομένων που απαιτούνται για την αποθήκευση και ορθή αναπαραγωγή του ήχου, ο οποίος ωστόσο συνεχίζει να ακούγεται σαν πιστή αναπαραγωγή του αρχικού ασυμπίεστου περιεχομένου από τους περισσότερους ακροατές. Εφευρέθηκε από μία ομάδα Γερμανών μηχανικών του Ιδρύματος Fraunhofer, εργαζομένων στα πλαίσια του προγράμματος EUREKA 147 DAB το οποίο έκανε έρευνα επάνω στο ψηφιακό ραδιόφωνο, και τυποποιήθηκε με βάση το πρότυπο ISO/IEC το 1991.

ΚαραλήςΓιώργος

4=Το MPEG-4 είναι μια μέθοδος ορισμού της συμπίεσης ψηφιακών δεδομένων ήχου και εικόνας (AV). Εισήχθη στα τέλη του 1998 και όρισε ένα πρότυπο για μια ομάδα μορφών κωδικοποίησης ήχου και εικόνας και σχετική τεχνολογία που συμφωνήθηκε από την ομάδα Κωδικοποίησης οπτικοακουστικών αντικειμένων. Οι χρήσεις του MPEG-4 περιλαμβάνουν συμπίεση δεδομένων AV για web και διανομή CD, με ήχο και βίντεο και εφαρμογές τηλεοπτικών εκπομπών.

5=Μορφή αρχείου ήχου Waveform (WAVE ή περισσότερο γνωστή ως WAV λόγω της επέκτασης του αρχείου (σπάνια, Audio for Windows)) είναι ένα πρότυπο μορφής αρχείου ήχου Microsoft και IBM για την αποθήκευση ενός bitstream ήχου σε Η / Υ . Πρόκειται για μια εφαρμογή της μεθόδου bitstream File Format Interchange File Format (RIFF) για την αποθήκευση δεδομένων σε "κομμάτια" και επομένως είναι επίσης κοντά στη μορφή 8SVX και AIFF που χρησιμοποιούνται στους υπολογιστές Amiga και Macintosh , αντίστοιχα. Είναι ο κύριος τύπος που χρησιμοποιείται στα συστήματα των Windows για ακατέργαστο και τυπικά μη συμπιεσμένο ήχο. Η συνηθισμένη κωδικοποίηση bitstream είναι η μορφή γραμμικού παλμικού κώδικα (LPCM).

6=Το MIDI (Musical Instrument Digital Interface, ελλ. Ψηφιακή Διασύνδεση Μουσικών Οργάνων) είναι ένα πρωτόκολλο που αποσκοπεί στην επικοινωνία και τον συγχρονισμό μεταξύ ηλεκτρονικών μουσικών οργάνων (όπως συνθετητές, ρυθμομηχανές, δειγματολήπτες, συσκευές χρονισμού), υπολογιστών και άλλων ηλεκτρονικών συσκευών, ανεξαρτήτως κατασκευαστή.

7=Το πρωτόκολλο MIDI δεν μεταδίδει ηχητικό σήμα, αλλά μηνύματα που περιέχουν πληροφορίες σχετικά με το τονικό ύψος και την ένταση μιας νότας, καθώς επίσης και σήμα χρονισμού που προσδιορίζει την ταχύτητα - το tempo/παλμό - ενός κομματιού.

8=Η μορφή αρχείου ανταλλαγής ήχου (AIFF) είναι ένα πρότυπο μορφής αρχείου ήχου που χρησιμοποιείται για την αποθήκευση δεδομένων ήχου για προσωπικούς υπολογιστές και άλλες ηλεκτρονικές συσκευές ήχου. Η μορφή αναπτύχθηκε από την Apple Inc. το 1988 βασισμένο στο IFF File Interchange File Format.
Τα δεδομένα ήχου στα περισσότερα αρχεία AIFF είναι μη συμπιεσμένη διαμόρφωση παλμικού κώδικα (PCM). Αυτός ο τύπος αρχείου AIFF χρησιμοποιεί πολύ περισσότερο χώρο σε δίσκο απ 'ό, τι οι μορφές απώλειας όπως MP3 - περίπου 10 MB για ένα λεπτό στερεοφωνικού ήχου.

2).Προετιμασία προγάμματος.

Βήμα 1
Κλασικά ανοίγω ένα Visual Studio 2010
Μετά πάω "File" -> "New" -> "Project..."
Επιλέγω πρότυπο " -> "Visual C#" -> "Windows για εφαρμογή "
Βήμα 2
Τώρα προσθέτω τον έλεγχο των Windows Media Player στην εργαλειοθήκη.
Δεξί κλικ στην εργαλειοθήκη
Επιλέγω "Επιλογή στοιχείων"
Κάνω αναζήτηση για το Windows Media Player μέσα στην καρτέλα "Συνιστώσες COM" : Τώρα ο έλεγχος του Windows Media Player προστίθεται στην εργαλειοθήκη.
Βήμα 3
Εδώ χρησιμοποιώ ένα πίνακα ελέγχου TableLayout στη φόρμα και ορίζω αυτές τις ιδιότητες σε "Columncount = 1, Dock = Fill και Rows = 3". Τώρα έχω εισάγει τον έλεγχο Windows Media Player από την εργαλειοθήκη στην πρώτη γραμμή του πίνακα TableLayout και ορίζω την ιδιότητα "Dock = Fill". Ένα πλαίσιο λίστας και κουμπί ελέγχου στη δεύτερη και τρίτη σειρά του πλαισίου

διαμόρφωσης πίνακα με ιδιότητα "Dock = Fill". Και ένα OpenFileDialog για να επιλέξω ένα ή περισσότερα αρχεία .mp3 για αναπαραγωγή.
Βήμα 4
Τώρα γράφω τον κώδικα C#

3).Επεξήγηση ορισμένων εντολών

"# - Δηλώνει τον προεπεξεργαστή.
<bitset> Παρέχει ένα κοντέινερ τύπου std::bitset, δηλαδή ένα πίνακα μπιτ (bit) - με λογικές τιμές τύπου bool (boolean)(tru, η false).
<deque>Παρέχει ένα κοντέινερ τύπου std::deque, δηλαδή μια ουρά στοιχείων όπου στοιχεία μπορούν να αφαιρεθούν και από την αρχή και από το τέλος.
<list> Παρέχει ένα κοντέινερ τύπου std::list, δηλαδή μια λίστα στοιχείων.
<map> Παρέχει ένα κοντέινερ τύπου std::map και ένα τύπου std::multimap, δηλαδή ένα ταξινομημένο πίνακα συσχετισμένων στοιχείων, και ένα πολλαπλό πίνακα συσχετισμένων(συγκεκρινόμενονων) στοιχείων.
<queue> Παρέχει ένα κοντέινερ τύπου std::queue, δηλαδή μια ουρά στοιχείων.
<set>Παρέχει ένα κοντέινερ τύπου std::set και τύπου std::multiset, δηλαδή ένα ταξινομημένο κοντέινερ συσχετισμένων(συγκεκρινόμενονων) στοιχείων.
<stack> Παρέχει ένα κοντέινερ τύπου std::stack, δηλαδή μια στοίβα στοιχείων.
<vector> Παρέχει ένα κοντέινερ τύπου std::vector, δηλαδή ένα δυναμικό πίνακα στοιχείων.
<algorithm> Είναι το αρχείο επικεφαλίδας για γενικούς αλγόριθμους που εφαρμόζονται σε κοντέινερ.
<iterator>
Είναι το αρχείο επικεφαλίδας για πρότυπα αντικειμένων-προσπελαστών (iterators). Οι προσπελαστές(δηλώνει δήναμη) χρησιμοποιούνται μέσα σε βρόχους με τους οποίους μπορούμε να προσπελάσουμε τα δεδομένα ενός κοντέινερ.
<memory>
Λειτουργίες για διαχείριση μνήμης στην C++, συμπεριλαμβάνεται και η κλάση std::auto_ptr.
<string>
Παρέχει το αντικείμενο-κλάση std::string με την λειτουργικότητα ενός αλφαριθμητικού.
<fstream>
Παρέχει τις δυνατότητες εισόδου/εξόδου μέσω κάποιου εξωτερικού αρχείου (ροές δεδομένων προς και από ένα εξωτερικό αρχείο).
<fstream>
Παρέχει τις δυνατότητες εισόδου/εξόδου μέσω κάποιου εξωτερικού αρχείου (ροές δεδομένων προς και από ένα εξωτερικό αρχείο).
<ios>
Παρέχει διάφορους τύπους δεδομένων και συναρτήσεις για την λειτουργία των ροών δεδομένων (iostreams).
<iostream>
Παρέχει την βασική είσοδο/έξοδο της C++ (ροές δεδομένων όπως διάβασμα από το πληκτρολόγιο μέσω της cin ή γράψιμο στην κονσόλα μέσω του cout).
<sstream>
Παρέχει την κλάση-πρότυπο std::sstream όπως και άλλες σχετικές κλάσεις-αντικείμενα για επεξεργασία αλφαριθμητικών (ροές δεδομένων αλφαριθμητικών).
<typeinfo>
Παρέχει λειτουργικότητες της C++ κατά την εκτέλεση-τρέξιμο του τελικού προγράμματος.

4).Επεξήγηση της γλώσσας C.

Επεξήγηση γλώσσας C++

Η C είναι μια ευέλικτη γλώσσα που μπόρει να χρησιμοποιηθεί για την ανάπτιξη σύνθετων εφαρμογών όπως αναπτιξη λειτουργικού συστήματος ενσωματωμένων συστημάτων. Είναι μια γλώσσα hardware.
Η C είναι μικρή γλώσσα με την έννοια ότι το συντακτικό της αποτελείται απο λίγες λέξεις με ειδική σημασία.
Ο κώδικας είναι μεταφέρσιμος. Δηλαδή το ίδιο πρόγραμμα μπορεί να λειτουργίσει σε Windows, Linux και Unix.
Η δημιουργεία του προγράμματος περιλαμβάνει την λειτουργία.
Συγγραφή του κώδικα - edit.
Μεταγλώτιση του - Compile
Σώνδεση του μεταγλωτισμένου κώδικα με το κώδικα συναρτήσεων βιβλιοθήκης - Link.
Η C ++ είναι μια γλώσσα προγραμματισμού. Κυριολεκτικά σημαίνει "αυξημένη C", αντανακλώντας τη φύση της ως εξέλιξη της γλώσσας C.
Η C ++ είναι μια απλή και ξεκάθαρη γλώσσα στις εκφράσεις της. Ενα κομμάτι κώδικα γραμμένο με C ++ μπορεί να θεωρηθεί από έναν ξένο προγραμματιστή λίγο πιο κρυφά από κάποιες άλλες γλώσσες λόγω της έντονης χρήσης ειδικών χαρακτήρων ({} [] * &! | ...), αλλά από τη στιγμή που κάποιος γνωρίζει τη σημασία τέτοιων χαρακτήρων, μπορεί να είναι ακόμα πιο σχηματική και σαφής από άλλες γλώσσες που βασίζονται περισσότερο στις αγγλικές λέξεις.
Επίσης, η απλοποίηση της διασύνδεσης εισόδου / εξόδου της C ++ σε σύγκριση με το C και η ενσωμάτωση της πρότυπης βιβλιοθήκης προτύπων στη γλώσσα καθιστά την επικοινωνία και τον χειρισμό δεδομένων σε ένα πρόγραμμα γραμμένο σε C ++ τόσο απλό όσο σε άλλες γλώσσες, χωρίς να χάσει τη δύναμη που προσφέρει.
Το OOP πρόκειται για ένα μοντέλο προγραμματισμού που αντιμετωπίζει τον προγραμματισμό από μια οπτική γωνία όπου κάθε στοιχείο θεωρείται αντικείμενο, με τις δικές του ιδιότητες και μεθόδους, αντικαθιστώντας ή συμπληρώνοντας το δομημένο πρότυπο προγραμματισμού, όπου εστιάστηκε στις διαδικασίες και τις παραμέτρους.

5).Ιστορία της γλώσσας c.

Η C++ αναπτύχθηκε από τον Bjame Stroustrup στα εργαστήρια Bell μεταξύ 1983-1985. Πριν το 1983 άρχισε με την γλώσσα προγραμματισμού Simula η οποια θεωρητε πρωτη χρονολογικά.
Πριν αναπτύξει την αντικειμενοστραφή γλώσσα προγραμματισμού C++, ανάπτηξε την πειραματική γλώσσα προγραμματισμού ¨C with Classes η οποία ανομάστηκε το 1983 C++.
Αρχικά η C++ χρησιμοποιήθηκε για την ανάπτυξη του λειτουργικού συστήματος UNIX. Οι συνεχείς βελτιώσεις και προσθήκες στην γλώσσα συνεχίστηκαν μέχρη το 1997. Το 1997 η C++ άρχισε να αποτελεί την κυριαρχεία/πρωτία σε γλώσσες προγραμματισμού.
Ο βασικός κορμός της C++ εινα λιτός. Η γλώσσα διαθέτει λίγες εντολές και μόνο τους βασικούς τύπους δεδομένων.
Ένα πρόγραμμα γραμμένο στην C με ελάχιστες η καθόλου αλλαγές. Μπορεί να μεταγλωτιστεί σε γλώσσα C++ χωρίς κανένα πρόβλημα.
Η γλώσσα C++ και Java αποτελούν τις πλέον δημοφιλείς και χαρακτηριστικές γλώσσες αντικειμενοστραφούς προγραμματισμού
Κατά την διάρκεια μετάφρασης της γλώσσας της C++, ο μεταγλωτιστής εντοπίζει τυχόν συνταχτικά λάθη που υπάρχουν στον πηγαίο κώδικα. Άμα ο μεταγλωτιστής δεν εντοπίσει κανένα λάθος, θα δημιουργήσει ένα αρχείο που περιέχει τον πηγαίο κώδικα. Το οποίο έχουμε την δηνατότητα να το τρέξουμε. Η C++ μπορεί να μεταφραστεί σχεδόν από όλους τους υπολογιστές και λειτουργηκά συστήματα. Το χαρακτηριστικό αυτό, σε συνδιασμό με τεράστιες δυνατότητες που περιέχει, την ευελιξία της και την απόδοση της, την καθιστούν σήμερα πρώτη επιλογή των προγραμματιστών και για την κατασκευή μεγάλων και απαιτητικών προγραμμάτων.

ΚαραλήςΓιώργος

Επεξήγηση γλώσσας C++

Τα δεδωμένα, οι ενδιάμεσες πράξεις αλλά και τα αποτελέσματα των προγραμμάτων αποθηκεύονται στην κεντρική μνήμη RAM.

6).Επεξηγηση της Ram.

"RAM"
Αν κοιτάξει κανείς τα χαρακτηριστικά ενός υπολογιστικού συστήματος, θα δεί την μνήμη RAM, η οποία αποτελεί βασικό κομμάτι αυτών των συσκευών και η χωρητικότητα της αυξάνεται συνεχώς (με την πρόοδο της τεχνολογίας).
Τα αρχικά της μνήμης RAM προέρχονται από το Random Access Memory (Μνήμη τυχαίας προσπέλασης), και σε αυτήν αποθηκεύονται δεδομένα και προγράμματα που πρόκειται να εκτελεστούν ή να υποστούν επεξεργασία. Λέγεται μνήμη τυχαίας προσπέλασης επειδή επιτρέπει την πρόσβαση στα δεδομένα της, ανεξάρτητα της θέσης στην οποία βρίσκονται. Τα δεδομένα μένουν αποθηκευμένα στην RAM για όσο χρόνο χρησιμοποιούνται από κάποιο λογισμικό ή από τον χρήστη. Μόλις περάσει αυτό το χρονικό διάστημα είτε αντικαθίστανται από άλλα δεδομένα, είτε διαγράφονται. Η μνήμη RAM «αδειάζει» μόλις απενεργοποιήσουμε τον υπολογιστή μας (ή όποια συσκευή διαθέτει RAM). Επομένως αν παραδείγματος χάριν υπάρχουν δεδομένα μέσα στην RAM και ο υπολογιστής απενεργοποιηθεί ή για κάποιο λόγο διακοπεί η τροφοδοσία του (από μπαταρία ή καλώδιο), τότε τα δεδομένα που υπάρχουν στην RAM εκείνη την στιγμή χάνονται.
Η χωρητικότητα της μνήμης RAM είναι πολύ σημαντική διότι όσο μεγαλύτερη είναι, τόσο πιο πολλα προγράμματα θα μπορούν να εκτελεσθούν ταυτόχρονα και έτσι η συσκευή μας θα είναι πιο γρήγορη. Οι υπολογιστές ξεκίνησαν με πολύ μικρές μνήμες RAM 12 Bytes – 3 Kbytes. Σήμερα έχουμε φτάσει στο σημείο να μιλάμε για 16 και 32 GB.

7).Μεταβλητες στην γλώσσα C.

Η C++ στον κώδικα της έχει μεταβλητές, οι οποίοι είναι μια θέση μνήμης την οποία έχουμε δεσμεύσει και της έχουμε δώσει όνομα. Η γλώσσα C++ υποστηρίζει διαφόρους τύπους μεταβλητών, οι βασικές μεταβλητές είναι 4.

8).Τύπος δεδομένων χαρακτήρα.

Ο ακαίρεος τύπος δεδομένων: Αναφέρεται σε δεδομένα που είναι ακαίρεοι αριθμοί. Στην C++ ο τύπος αυτός συμβολίζεται με int η long int.
Ο πραγματικός τύπος δεδομένων: Αναφέρεται σε δεδομένα που είναι ακαίρεοι αριθμοί κίνητης υποδιαστολής με μικρή η μεγάλη ακρίβεια. Στην C++ ο τύπος αυτός συμβολίζεται με float η double.
Ο τύπος δεδομένων χαρακτήρα: Αναφέρεται σε μεμονομένους χαρακτήρες. Στην C++.Ο τύπος αυτός σύμβολίζεται με char.

9).Τύπος δεδομένων συνόλου χαρακτήρα.

Ο τύπος δεδομένων συνόλου χαρακτήρων. Αναφέρεται σε σύνολα χαρακτήρων. Η C++ δεν υποστηρίζει άμεσα αυτόν τον τύπο δεδομένων και διαχερίζεται διαφορετικα τους χαρακτήρες. Όμως με την χρήση της προκαθορισμένης βιβλιοθήκης πρότυπων της C++ υπάρχει μια έμμεση υποστήριξη του συνόλου χαρακτήρων μέσω της κλάσης string
Ο λογικός τύπος δεδομένων. Αναφέρεται σε δεδομένα με τιμή αλήθεια η ψέμα(boolean). Στη C++ ο τύπος αυτός σύμβολίζεται με bool.

ΚαραλήςΓιώργος

Επεξήγηση γλώσσας C++

Οι σταθερές είναι οι προκαθορισμένες τιμές που δεν μεταβάλονται/αλλάζουν κατά την διάρκεια της εκτέλεσης ενός προγράμματος. Υπάρχουν αντίστοιχες σταθερές για κάθε τύπο δεδομένων.
Οι εντολές είναι μια φράση της γλώσσας προγραμματσμού που αναγκάζει τον Η/Υ να εκτελέσει μια συγκεκριμένη λειτουργία.
Οι περισσότερες γλώσσες ενθαρύνουν τον προγραμματιστή να χωρίσει το πρόγραμμα σε μικρότερα τμήματα, τα οποία ονομάζονται υποπρογράμματα. Ένα τμήμα προγράμματος (υποπρόγραμμα) στην C++ ονομάζεται συνάρτηση fuction π.χ void.
Αναγνωριστηκά καλούνται τα ονόματα που χρησιμοποιεί μια γλώσσα για να αναφερθεί στα επιμέρους στοιχεία της. Κάθε γλώσσα έχει δικούς της κανόνες για την σύνταξη των αναγνωριστικών της. Οι κανόνες αυτοί καθορίζουν το πλήθος και το είδος των επιτρεπτών χαρακτήρων.

10).Μη δεδομένος προγραμματίσμός:Saggetti programs.

Μη δεδομένος προγραμματίσμός:Saggetti programs: Συνήθως όταν ξεκινάει κάνεις να μαθαίνει προγραμματισμό, τα πρώτα προγράμματα που κατασκευάζει είναι μικρά και αποτελούνται από ένα μοναδικό εννιαίο σύνολο εντολών. Όλα τα δεδομένα είναι διαθέσημα στο πρόγραμμα το οποίο έχει πλήρη προς βάση στα δεδομένα από κάθε του σημείο, ακόμα από τα σημεία που δεν χρειάζεται να έχουν πρόσβαση σε συγκεκριμένα δεδομένα. Ο μη δομημένος προγραμματισμός οδηγεί σε δυσανάγνωστα προγράμματα με μεγάλο κόστος συντήρησης , με μηδενικές δυνατότητες επαναχρησιμοποίησης και επέκτασης.

11).Δομημένος διαδικαστικός προγραμματισμός.

Στον δομημένο διαδικαστικό προγραμματισμό το πρόγραμμα χωρίζεται πλέον σε ξεχωριστές ενότητες, κάθε μια από τις οποίες αποτελεί μία ξεχωριστή λειτουργία του προγράμματος. Μία από αυτές τις διαδικασίες αποτελεί το κυρίως προγραμμα main. Ο δομημένος διαδικαστικός προγραμματισμός επιτρέπει την αφαιρετικότητα δηλαδή την απόκρυψη δεδομένων από τα διάφορα τμήματα του προγράμματος.
Μια διαδικασία μπορεί να έχει τα δικά της τοπικά δεδομένα στα οποία εχει πρόσβαση μόνο η ίδια, ενώ στα καθολικά δεδομένα έχουν άμεση πρόσβαση όλες οι διαδικασίες του προγράμματος.
Το κυρίως πρόγραμμα η μια διαδικασία, όταν καλεί μια άλλη διαδικασία, έχει την δυνατότητα να της μεταβιβάσει δεδομένα (στα οποία δεν εχει πρόσβαση) κατά την ώρα της κλήση της. Επίσης μια διαδικασία μπορεί να επιστρέψει τα επεξεργασμένα δεδομένα στο πρόγραμμα που την κάλεσε.

12).Τμηματικός προγραμματισμός.

Στον τμηματικό προγραμματισμό, το πρόγραμμα χωρίζεται σε ξεχωριστά τμήματα, τα οποία βρήσκονται σε διαφορετηκά αρχεία πηγαίου κώδικα και κάθε ένα μπορεί να περιέχει πολλές διαδικασίες. Κάθε τμήμα έχει τα δικά του καθολικά δεδομένα στα οποία έχουν πρόσβαση μόνο οι διαδικασίες του συγκεκριμένου προγράμματος.
Με αυτόν τον τρόπο έχουμε μεγαλύτερο βαθμό αφαιρετηκότητας και τα δεδομένα είναι προσβάσιμα μόνο από τις διαδικασίες που τα χρειάζονται.
Τόσο στον δομημένο προγραμματισμό και στον τμηματικό προγραμματισμό τα δεδομένα και η διαδηκασίες αποτελούν διαφορετηκές οντότητες. Οι διαδικασίες πρέπει να έχουν πρόσβαση στα δεδομένα, όλα τα δεδομένα θα πρέπει ταυτόχρονα να είναι προφυλαγμένα από μη-εξουσιοδοτημένη χρήση. Η αφαιρετικότητα βοηθάει αλλά δεν λύνει τα προβλήματα του διαχωρισμού δεδομένων και διαδικασιών.

ΚαραλήςΓιώργος

Επεξήγηση γλώσσας C++

13).Αντικειμενοστραφής προγραμματισμός.

-Αντικειμενοστραφής προγραμματισμός: Ο Αντικειμενοστραφής προγραμματισμός συγκεντρώνει όλα τα στοιχεία του δομημένου και τμηματίκου προγραμματισμού, καταργόντας το χάσμα που υπάρχει μεταξύ δεδομένων και διαδικασιών, εισάγοντας παράλληλα νέες έννοιες και μια διαφορετική φιλοσοφία όσον αφορά στο σχεδιασμό και στην ανάπτυξη των προγραμμάτων.
Στον αντικειμενοστραφή προγραμματισμό τα δεδομένα μας είναι εκτεθημένα σε ότι έχει πρόσβαση στους αποθηκευτηκούς χώρους. Με αυτόν τον τρόπο μπορεί να γίνει **ολίγιστη** χρήση των δεδομένων ακόμη και για διαφορετηκό σκοπό.
Η επεξεργασία των δεδομένων δεν είναι αυτοματοποιημένη και ο καθένας μπορεί να ακολουθήσει μια διαφορετική διαδικασία.
Αν αλλάξει ο χώρος που αποθηκεύουμε τα δεδομένα. Πρέπει να αλλάξουμε και την επεξεργασία ώστε να χρεισημοποιούμε άλλο αποθηκευτηκό χώρο.
Πάντα αποθηκεύουμε το αρχείο μας με την κατάληξη .c και πάντα τα αποθηκεύουμε στον ίδιο φάκελο για να μην τα ψάχνουμε(για να τα βρήσκουμε μπα μπαμ).
ΟΙ κλάσεις και τα αντικείμενα είναι έννιες στις οποίες βασίζεται όλη η φιλοσοφία και τα χαρακτηριστικά του αντικοιμενοστραφούς προγραμματισμού γενικότερα η κλάση είναι μια ένοια η οποία προσδιορίζει μια κατηγορία αντικειμένων και ταυτόχρονα περιγράφει τα χαρακτηριστικά τους.
Ένα αντοκείμενο μιας κλάσης έχει όλα τα χαρακτηριστικά και τις λειτουργείες κλάσης της οποίας ανήκει. Ένα αντικείμενο, λέμε ότι αποτελεί ένα στοιγμιότυπο μιας κλάσης.

14).Εθνυλάκωση.

Χαρακτηριστικά και οι λειτουργείες ενός αντικειμένου μπορεί να είναι εκτεθημένα στον χρήστη ώστε να μπορεί άμεσα να αλλάξει τις τιμές τους η να χρεισημοποιήσει τις λειτουργείες του. Μπορεί δηλαδή να αλλάξει την ποσότητα του΄΄Νερού΄΄ και να αλλάξει την λειτουργεία του ΄΄ελένχου΄΄.
Η χρήση των κλάσεων και των αντικοιμένων δίνει επίσης την δυνατότητα να αποκρύψουμε μερικά από τα χαρακτιριστικά και τις λειτουργίες ενός αντικειμένου ώστε ο χρήστης να μην μπορεί να έχει πρόσβαση σε αυτά.
Γενικότερα η ενθυλάκωση δίνει την δυνατότητα στις γλώσσες αντικοιμενοστραφούς προγραμματισμού να ομαδοποιούν και να αποκρύπτουν δεδομένα και διαδικασίες των αντικειμένων.

15).Κληρονομικότητα.

-Η κληρονομικότητα είναι από τα βασικά χαρακτιριστικά του αντικειμενοστραφούς προγραμματισμού. Μια κλάση μπορεί να παράγεται από μια άλλη βασική κλάση και ταυτόχρονα να κληρωνομει τα χαρακτηριστικά και τις λειτουργείες της. Στην νέα κλάση μπορεί να οριστούν επιπλέων χαρακτηριστικά και λειτουργείες.
Γενικότερα η κληρονομικότητα είναι η δυνατότητα παραγωγής μιας νέας κλάσης από μια υπάρχουσα βασική κλάση. Η νεα κλάση (παραγωγηκή) κληρονωμεί όλα τα τα χαρακτηριστικά και τις λειτουργείες της βασικής κλάσης όλα ταυτόχρονα μπορεί να ορίσει δικά της επιπλέων χαρακτηριστικά και επιπλέων λειτουργείες.

16).Πολυμορφισμός.

Επεξήγηση γλώσσας C++

-Ο πολυμορφισμός είναι η δυνατότητα που περιέχουν οι αντοικειμενοστραφείς γλώσσες προγραμματισμού στα αντικείμενα ώστε να συμπεριφέρονται διαφορετηκά.

17).Υπερφόρτωση τελεστών.

Η υπερφόρτωση τελεστών: Είναι ένα ακόμη χαρακτηριστικό του πολυμορφησμού.
Το λογικό διάγραμμα ενός προβλήματος δεν καθορίζεται μονόσημαντα. Μπορεί να είναι εντελώς από επιγραμματικό ως πολύ αναλητηκό ανάλογα με την χρήση για την οποία προσδηορίζεται.

18).Γλώσσα UML.

-Η γλώσσα "UML" εχει ιδιέταιρη εφαρμογή στην ανάπτυξη και μοντελοποίηση αντικειμενοστραφών εφαρμογών. Η UML (United Modeling Language) είναι μια γλώσσα μοντελοποίησης, η οποία μέσω συγκεκριμένων στοιχείων και διαγραμμάτων μας δίνει την δυνατότητα να απικονίσουμαι πολύπλοκες εφαρμογές.
ενοποιημένη γλώσσα σχεδιασμού (unified modeling language) (UML) είναι μια γραφική γλώσσα για την οπτική παράσταση, τη διαμόρφωση προδιαγραφών και την τεκμηρίωση συστημάτων που βασίζονται σε λογισμικό. Η UML στοχεύει στο σχεδιασμό αντικειμενοστρεφών συστημάτων.
Το σχέδιο είναι μια απλοποιημένη παράσταση της πραγματικότητας.
Σχεδιάζουμε για να μπορέσουμε να καταλάβουμε το σύστημα που αναπτύσσουμε. Έτσι δημιουργώντας ένα σχέδιο επιτυγχάνουμε τέσσερεις στόχους:
Παριστάνουμε οπτικά το σύστημα που έχουμε ή θέλουμε να κατασκευάσουμε,
προσδιορίζουμε τη δομή και τη συμπεριφορά του συστήματος,
δημιουργούμε ένα πρότυπο για να βασίσουμε την κατασκευή του συστήματος,
τεκμηριώνουμε τις αποφάσεις που λάβαμε.
Σε όλους τους τεχνολογικούς τομείς ο σχεδιασμός βασίζεται σε τέσσερεις βασικές αρχές:
η επιλογή του είδους του σχεδίου έχει επίπτωση στον τρόπο και την μορφή επίλυσης του προβλήματος,
όλα τα σχέδια εκφράζονται σε διαφορετικές βαθμίδες ακρίβειας,
τα καλύτερα σχέδια σχετίζονται με την πραγματικότητα,
ένα είδος σχεδίων δεν είναι ποτέ αρκετό.
Η UML περιλαμβανει τρία βασικά στοιχεία:
Οντότητες
Σχέσεις
Διαγράμματα

Η UML είναι μια πλήρης και πλούσια γλώσσα με εξαιρετικά ευρύ πεδίο εφαρμογής. Στο μάθημα αυτό θα εξετάσουμε εξαιρετικά συνοπτικά τον τρόπο παράστασης ορισμένων αντικειμενοστρεφών δομών σε UML.
Διαγράμματα
Η UML ορίζει τα παρακάτω διαγράμματα:

19).Διαγράμματα συμπεριφοράς.

Διάγραμμα περιπτώσεων χρήσης (use case diagram)
Διαγράμματα δομής
Διάγραμμα κλάσεων (class diagram)
Διάγραμμα αντικειμένων (object diagram)
Διαγράμματα συμπεριφοράς

ΚαραλήςΓιώργος

Επεξήγηση γλώσσας C++

Διάγραμμα καταστάσεων (statechart diagram)
Διάγραμμα δραστηριοτήτων (activity diagram)
Διαγράμματα αλληλεπίδρασης
Διάγραμμα ακολουθίας (sequence diagram)
Διάγραμμα συνεργασίας (collaboration diagram)
Διαγράμματα δομής υλοποίησης
Διάγραμμα εξαρτημάτων (component diagram)
Διάγραμμα ανάπτυξης (deployment diagram)
1= Η C++ είναι μια αντικειμενοστραφής μεταγλωτιζομένη γλώσσα τα χαρακτηριστικά της οποίας ευνοούν τον τμηματικό προγραμματισμό.
2= Στη C++ αρκετές λειτουργείες υλοποιούνται από τα αντικείμενα και στις συναρτήσεις βιβλιοθήκης στην οποία την συνοδεύουν.
3= Τα διαφορετηκά είδη δεδομένων που μπορούν να χειριστούν μια γλώσσα ονομάζονται τύποι δεδομμένων.
4= Μια μεταβλητή είναι το όνομα μιας θέσης μνήμης της οποίας η τιμή μπορεί να μεταβάλεται
5= Ο αντικειμενοστραφής προγραμματισμός βασίζεται στην μοντελοποίηση του προβλήματος χρεισημοποιόντας αντικείμενα.
6= Κάθε αντικείμενο ανήκει σε μια κλάση η οποία προσδιορίζει τα χαρακτηριστικά και τις λειτουργίες του.
7= Τα κύρια χαρακτηριστικά του αντικειμενοστραφούς προγραμματισμού είναι η ενθυλάκωση, η κληρονομικότητα και ο πολυμορφησμός.
8= Το λογικό διάγραμμα είναι ένας οπτικός τρόπος παρουσίασης μιας διαδικασίας η ενός προγράμματος και συνβολίζει την κατανώηση του. Η UML, αντίστοιχα είναι μια γλώσσα μοντελοποίησης, η οποία έχει εφαρμογή στον σχεδιασμό περίπλοκων αντικειμενοστραφών εφαρμογών.

20).Εκτελέσημες προτάσεις.

Οι εκτελέσιμες προτάσεις αποτελούνται από εντολές παραστάσεις, κλήσης συναρτήσεων και χρήση αντικείμενων, αποτέλουν μια συγκεκρημένη λειτουργία.
Κάθε Δηλωτική πρόταση και εκτελεσίμη πρόταση τερματίζεται με Ελληνικό ερωτηματικό ;
Μπορούμε στο πρόγραμμα μας να χρεισημοποιήσουμε την fuction () η οποία δεν έχει στανταρ όνομα, είναι μια δηλωτική πρόταση.
Σε κάθε συνάρτηση μπορεί να υπάρχουν δηλωτηκές προτάσεις για μεταβλητές. Οι μεταβλητές αυτές ονομάζονται τοπικές μεταβλητές και μπορούν να χρησημοποιηθούν μόνο μέσα στις συναρτήσεις στις οποίες δηλώθηκαν:
Ο σωστός σχεδιασμός προγράμματος επιβάλει την χρήση σχολίων μέσα στο ίδιο το πρόγραμμα με σκοπό να κάνει το πρόγραμμα περισότερο κατανοήτο. Χρησημοποιόντας
/* Για αρχή του σχολίου και */ για κλείσημο και τα σχόλια δεν επιρεάζουν το πρόγραμμα.
Η αλιώς χρησημοποιούμε δυό // για σχόλια τα οποία τα γράφουμε μόνο στην αρχή.
Επειδή το conio.h δεν αποτελεί μέρος του προτύπου C. Πρόκειται για επέκταση Borland και λειτουργεί μόνο με τους μεταγλωττιστές Borland (και ίσως με μερικούς άλλους εμπορικούς μεταγλωττιστές). Η Dev-C ++ χρησιμοποιεί το GCC, τη συλλογή του GNU Compiler, όπως είναι ο μεταγλωττιστής. Το GCC είναι αρχικά ένας μεταγλωττιστής UNIX και έχει ως στόχο τη φορητότητα και τη συμμόρφωση με τα πρότυπα.
Αν πραγματικά δεν μπορείτε να ζήσετε χωρίς αυτούς, μπορείτε να χρησιμοποιήσετε τις λειτουργίες Borland με αυτό τον τρόπο:
Συμπεριλάβετε το conio.h στην πηγή σας και προσθέστε το C: \ Dev-C ++ \ Lib \ conio.o στις "Επιλογές Linker" στις επιλογές του Project (όπου C: \ Dev-C ++ είναι όπου εγκαταστήσατε το Dev-C ++).
Παρακαλώ σημειώστε ότι η υποστήριξη conio δεν είναι τέλεια

ΚαραλήςΓιώργος

```
#include<iostream>
#incluld<windows.h>
#include<System.h”
Using namespace std;
Int main()
{
playSound(TEXT (“music.wav”), NULL, SNO_SYNC);
system(“pause”);
return 0;
}
!
```

Έκανα καινούργιο φάκελο, μέσα σε αυτό αποθήκευσα το πρόγραμμα και το τραγούδι.wav.

21).Εκδοση Linux.

Υπήρξε μια έκδοση του Linux, αλλά εγκαταλείφθηκε, κυρίως επειδή το Dev-C ++ είναι γραμμένο στους Δελφούς, αλλά η έκδοση Linux του Delphi (Kylix) δεν ήταν τόσο ελπιδοφόρες όσο θα έπρεπε. Αλλά υπάρχουν εξαιρετικές εναλλακτικές IDE για το Linux, όπως το KDevelop και το Anjuta. Αν έχω να επιλέξω μεταξύ αυτών των δύο, θα επέλεγα την Anjuta.

22).Επεξήγηση στης βιβλιοθήκες της C++.

#include <iostream> = Η πρόταση αυτή αναγκάζει τον υπολογιστή να περιλαμβάνει το αρχείο iostream κατά την διαδικασία της μεταγλώτησης. Το αρχείο αυτό περιέχει της δηλώσεις των αντικειμένων και των συναρτήσεων εισόδου/εξόδου της C++. Για να μπορέσουμε να χρεισημοποιήσουμε οποιαδήποτε από τις λειτουργίες εισόδου/εξόδου στο πρόγραμμα μας.
Using namespace std; = Η πρόταση αυτή ενημερώνει των μεταγλωτηστή της C++, ότι τα ονόματα θα ανήκουν στο όνομα χώρου std. Η χρήση ονομάτων χώρου δίνει την δεινατότητα στην C++ να χρησημοποιεί οντότητες που έχουν το ίδιο όνομα, αλλά διαφορετική λειτουργεία ανάλογα με το όνομα χώρου στο οποίο ανήκουν.

23).Η συνάρτηση main().

Η συνάρτηση main() είναι υποχρεωτική σε κάθε πρόγραμμα της C++. Στην περίπτωση που το πρόγραμμα μας έχει μόνο μια συνάρτηση, αυτή πρέπει να είναι η main(). Αν το πρόγραμμα μας εχει περισσότερες συναρτήσοις, η main πρέπει να είναι η πρώτη, πάνω πάνω.
Το αντικείμενο cout χρησημοποιήτε για την έξοδο πληροφοριών στην οθόνη.
Τα κενά διαστήματα και οι αλλαγές γραμμής δεν λαμβάνονται υπόψη από τον μεταγλωτηστή της C++.
Ένα απλό πρόγραμμα στην C++ αποτελείτε συνήθως από 3 βασικά τμήματα.ενθηλακωση,τμηματισμο και πολυμορφησμο…
Μια συνάρτηση με το όνομα main(): Αυτή η συνάρτηση είναι αυτή που καλήτε και εκτελείται πρώτη.
Οι προτάσεις μιας συνάρτησης μπορεί να είναι δυο είδων, δηλωτίκες και εκτελέσημες.
Μια δηλωτική πρόταση δηλώνει την ύπαρξη μιας οντότητας η οποία μπορεί αργότερα να χρησημοποιηθεί από τον κώδικα του προγράμματος. Όλες οι μεταβλητές που χρησημοποιούνται σε ένα πρόγραμμα της C++ πρέπει από την αρχή να έχουν δηλωθεί. Μια δήλωση μεταβλητής περιλαμβάνει το όνομα της μεταβλητής και τον τύπο της.

24).Μερικές εντολές και οι λειτουργείες τους.

Επεξήγηση γλώσσας C++

#- Το σύμβολο αυτό σημαίνει ότι τη γραμμή αυτή τη διαχειρίζεται ο προ-επεξεργαστής.
{ = Η αριστερή αγκίλη σηματοδοτεί την αρχή των προτάσεων της συνάρτησης
} = Η δεξιά αγκίλη σηματοδοτεί το τέλος των προτάσεων της συνάρτησης.

#include <stdio.h>-Δίνει την οδηγία να συμπεριληφθεί στο πρόγραμμα το αρχείο κεφαλίδας.
{}-Οτιδήποτε βρίσκεται ανάμεσα στο ανοιχτό άγκιστρο και το κλειστό άγκιστρο είναι το κυρίως πρόγραμμα.
\n- Αλλαγή γραμμής.
int main()-Πρέπει επίσης να συμπεριλαμβάνεται απαραίτητα σε κάθε μας πρόγραμμα. Η εντολή αυτή ορίζει την έναρξη της συνάρτησης main που είναι τύπου int.(intenger)
printf( )-Είναι μία εντολή εξόδου. Τυπώνει το περιεχόμενο της στην οθόνη.
double a1;-Δεσμεύει στην μνήμη του υπολογιστή ένα σύνολο από θέσεις 8 διαδοχικές θέσεις μνήμης (8 bytes) που τις ονομάζει a1 και είναι τύπου πραγματικού αριθμού (δηλαδή αριθμού με δεκαδικό μέρος) διπλής όμως ακρίβειας.
float a1;- Δεσμεύει στην μνήμη του υπολογιστή ένα σύνολο από θέσεις 8 διαδοχικές θέσεις μνήμης (4 bytes) που τις ονομάζει a1 και είναι τύπου πραγματικού αριθμού (δηλαδή αριθμού με δεκαδικό μέρος) διπλής όμως ακρίβειας.

\f	Αρχίζει μία νέα οθόνη η σελίδα
\n	Μετακινείται στην αρχή της επόμενης γραμμής
\r	Μετακινείται στην αρχή της τρέχουσας γραμμής
\t	Μετακινείται στην επόμενη θέση στήλη (tab)
+	Πρόσθεση
-	Αφαίρεση
*	Πολλαπλασιασμός
/	Διαίρεση
%	Υπόλιπο ακαίρεας τιμής

{ } // Χρεισημοποιούνται για το που αρχίζει μια μεταβλητή και το χώρο που βρήσκεται.
_ // Το Underscore είναι έγκυρος χαρακτήρας σε αναγνωριστικά C ++.
int main() // // Βασικότατη για να γράψουμε ένα πρόγραμμα, βασικά το δηλώνει.
int; // Δηλώνει μεταβλητή ακαίρεου
( ) // Χρεισημοποιουνται για προτερεότητα πράξεων.
ifstream // Γιατι το
ofstream ofNames; // Κατασκευάζει ένα αντικείμενό. κάνοντας μια λίστα με ονόματα.
ifstream ifNames; // Εξάγει χαρακτήρες από τη ροή, ως μη μορφοποιημένη είσοδος. κάνοντας μια λίστα με ονόματα
-- // Ο χειριστής υποδιαστολής - είναι πανομοιότυπος με τον διαχειριστή προσαυξήσεως ++, εκτός από το ότι μειώνει την τιμή κατά ένα αντί να αυξάνει την τιμή κατά ένα.
pos=0, // Μεταβλητη
DataLen=0// Έτιμη συνάρτηση που εσύ της έχεις ορίσει τι να κάνει
NumOfLines- Ο αριθμός των γραμών
STRLen; // Το strlen () παίρνει μια μηδενική τερματισμένη byte str string ως όρισμα και επιστρέφει το μήκος της. Το μήκος δεν περιλαμβάνει τον μηδενικό χαρακτήρα. Εάν δεν υπάρχει κανένας μηδενικός χαρακτήρας στη συμβολοσειρά, η συμπεριφορά της λειτουργίας είναι απροσδιόριστη.
void ArrayStore // Μια διάταξη είναι μια σειρά στοιχείων του ίδιου τύπου τοποθετημένα σε συνεχείς θέσεις μνήμης που μπορούν να αναφερθούν μεμονωμένα προσθέτοντας ένα ευρετήριο σε ένα μοναδικό αναγνωριστικό.
++;// Χρεισημοποιήτε για να προστέσει άλο ένα.
void // Το void χρεισημοποιήτε για να μην επιστρέψει τιποτα

ΚαραλήςΓιώργος

Επεξήγηση γλώσσας C++

else if(SavedText[0] == ' '){// Εάν δηλαδη το πρώτο αντικοίμενο της λίστας είναι αποθηκευμένο, έχω το κενό
Str // Χρεισημοπειοίτε για να επιστρέψει.
hwnd // Με αυτό δηλώνω κουμπιά
ifRead me1.open("Data/Read me.txt");// Εάν διαβάσω το me1.open θα ανοίξει το κοίμενο απο το Read me
if (ifRead me.fail()){// Εάν το κείμενο αποτύχει.
cerr // Τυπική ροή εξόδου για σφάλματα.
cout/prindf // Εμφανίζει στο μα΄΄υρο παραθυράκι.
exit(1); // Κάνει έξοδο στην μεταβλητή 1
STR// Αυτό είναι ένα stream. Το stream είναι ένας δυαμεσολαβητής μεταξύ των συσκευών. WS // Εξάγει όσο το δυνατόν περισσότερους χαρακτήρες από την τρέχουσα θέση στην ακολουθία εισόδου. Η εξαγωγή σταματά αμέσως μόλις εντοπιστεί ένας χαρακτήρας που δεν είναι κενός. Αυτοί οι εξαγόμενοι χαρακτήρες λευκού χρώματος απορρίπτονται.
WM // Ορισμός μηνήματος
HANDLE // Μια λαβή μπορεί να είναι χρήσιμη για την αποθήκευση καταστάσεων (μεταξύ άλλων).
// // Κάνω τα σχόλια.
cout << "Now you will see a red text" // Μας δείχνει το κείμενο με κόκκινο χρώμα.
For(int i=0; i < 10; i +=2) // Θα μας χρωματίσει τα κείμενα απο το 0=πρώτο κείμενο μέχρι το 10ο κείμενο
exit(1); // Κάνε έξοδο
system("pause"); // Το πρόγραμμα σταματάει.
return 0; // Το πρόγραμμα ξαναρχίζει απο την αρχή.
#include <stdlib.h> // Βοηθάει στην αλλαγή ενος τύπου στην μεταβλήτη. ----------
#include"MMSystem.h" // Αυτό παίζει την μουσική.
#include <stdio.h> // Είναι μία δήλωση που λέει στον μεταγλωτιστή να εισάγει περιεχόμενα.
#include<graphics.h> // Χρησιμοποιεί βιβλιοθήκες για εμφάνιση 2d και 3d περιεχομένου.
#include<conio.h> // Αντιπροσωπεύει το Αρχείο κεφαλίδας εισαγωγής, το οποίο διαχειρίζεται είσοδο/έξοδο.
//280-358 δημιουργία τετραγωνακίων + 529-552 Δήλωση φακέλου.
Printf // Εμφανίζει στην οθόνη.
std::string filename = "Songs"; // Δημιουργεί ένα φάκελο.
using namespace std; // Αυτή η πρόταση περιλαμβάνει την είσοδο και έξοδο και άλλες εντολές.
Pow-Ύψωση του double a στο double b = Pow(a , b) (χρειάζεται το #include της math.h)
ab
sizeof()-Αυτή η συνάρτηση δίνει το μέγεθος του χώρου μνήμης που παρακρατείται μετά την δήλωση μεταβλητών τύπου short int, int, long int
;- Στο τέλος κάθε εντολής βάζουμε πάντα ελληνικό ερωτηματικό.
return 0-Επιστρέφει την ακέραια τιμή 0 και σηματοδοτεί το τέλος της συνάρτησης main.
void main()με συνδιασμό το return ;-Δεν επιστρέφει τίποτα
&-Διαίρεση
&-πηλίκο
int input; // Η εντολή αυτή προσθέτει κείμενο στο πρόγραμμα το οποίο θα το διαβάζει εκτός και απο τον προγραμματιστή και ο χρήστης.
fstream textfile; // προσθέτει τον φάκελλο που θα υπάρχει στο πρόγραμμα.
// Στην αναζήτηση (ανάκτηση) επίσης δεν θέλουμε χώρους. συνεπώς bool Search = true αν θέλουμε να πραγματοποιήσουμε αναζήτηση (ανάκτηση).
textfile.open("Bolivar.txt");// προσθέτει τον φάκελλο που θα υπάρχει στο πρόγραμμα.
textfile.close();// Καταχωρεί τον φάκελλο στο πρόγραμμα.
cout << " zavarakatranemia<< end1;// Εμφανίζει στο πρόγραμμα το ζαβαρακατρανέμια.

ΚαραλήςΓιώργος

Επεξήγηση γλώσσας C++

return 0; // Το ξαναρχίζει από την άρχη.
PlaySound(TEXT ("antegeia.wav"),NULL,SND_SYNC); // Εδώ ορίζει το τραγουδι.
NULL avoid memory problems and works on the aplycasion.
//SYNC_ is calling to sychroniozed the option flow with the control system flow.
system("pause"); Σταματάει το σύστημα.
}using namespace std;//Το std ε.ιναι το πρότυπο και πολλά άλλα πράγματα ορίζονται σε αυτό αυτό σημαίνει ότι ένας τρόπος για να τους καλέσετε είναι να χρησιμοποιήσετε std
HWND NAMEBOX, Read me,G7Red,G710_2_1925,Playlistname,Playlist, SEARCH, RESETBUTTON, Image, Play,stop,next,previus; // Ορίζει τα αντικείμενα που μπαίνουν πάνω στο παράθυρο
--char SavedText[50],*FixedData, CharSave[3], c; // Το κείμενο αποθηκέυει 50 γραμές,κείμενα.
--SavedText[50] // Για να φτιάξω πίνακα.
ifstream ifWrite me; // Για να γράψω το αρχείο.
ofRead me.close(); // Το παραράθυρο Read me κλείνει.
//Δημιουργώντας μια λειτουργία Single-Array-Store που θα είναι δεκτή κάθε νέος "SUBMITted" πελάτη custumerNAME στον πίνακα.
FixedData[i]=SavedText[i]; //Φτιάχνει τον φάκελο μέσω του"for()"
ArrayStore(FileName, FixedData);//Επίσης αποθηκεύει την διάταξη διαθέσιμη στην search.
else if(Search==true) { //Εάν πραγματοποιούμε αναζήτηση, αποθηκεύστε τη σταθερή λέξη που αναζητήσατε στο "FixedData []", το οποίο θα χρησιμοποιηθεί στο πρόγραμμα για αναζήτηση αργότερα.Για την αποστολή αρχείων στο SpaceCheckRemove () Και στη συνέχεια στο DataStore ().
void WriteData(HWND BoxName, string FileName){ //Δεν θα μπορούσε να χρησιμοποιήσει "instream FileName" λόγω λάθους επίσης μετακινει καθε επιλεγμενο txt μεσα απο διατάξεις. απότε μπορούν να χρησημοποιηθούν εύκολα στην αναζήτηση.
Η πρώτη θέση [0] κάθε πίνακα είναι ο αριθμός των θέσεων αυτού του πίνακα.
i=1;//Σημειώστε ότι αυτό αρχίζει από το [1].
STRNAMES=new string[NumOfLines+100]; //Χρειάζετε να δημιουργήσει την διάταξη και στις δύο περιπτώσεις επειδή χρειαζόμαστε να δημοουργίσουμε την διάταξη και στις δυο περιπτώσεις επειδή χρειάζετε να αποθηκεύσουμε τα ονόματα εφώσον το OFSP θα τρέχει.
NAMEBOX //Εδώ είναι τα ονόματα των κουτιών.
ifRead me.close(); // Άμα το κουτάκι Read me κλήσει.
SetMenu(hwnd,hMenubar);// Έδω δηλώνω τα κουτάκια στο παράθυρο.
NAMEBOX= CreateWindow(TEXT("edit"), TEXT("Read me"), // Εδώ φτιάχνεται το τετραγωνάκι // WS_VISIBLE | WS_CHILD | WS_BORDER | ES_AUTOHSCROLL, // Εντολές για την εμφάνιση για το τετραγωνάκι.
10, 10, 200, 30, // χ,ψ, οριζόντιο,κάθετο.
GK8_2_1981= CreateWindow(TEXT("edit"), TEXT("Red"), // Εδω φτιαχνετε το τρετραγωνακι αριθμου τηλ //
WS_VISIBLE | WS_CHILD | WS_BORDER | ES_AUTOHSCROLL, // Εντολες για ενφανιση // 220, 10, 200, 30,
GK10_2_1925= CreateWindow(TEXT("edit"), TEXT(""), // Δευτερος αριθμος //
SEARCH= CreateWindow(TEXT("edit"), TEXT("SEARCH"), // Καποια παραγματα για τον πελατι // WS_VISIBLE | WS_CHILD | WS_BORDER | ES_AUTOHSCROLL,
SEARCHBUTTON= CreateWindow(TEXT("button"), TEXT("SUBMIT"), // Γραφει το τραγουδι WS_VISIBLE | WS_CHILD | WS_BORDER | ES_AUTOHSCROLL,// Γιατι το
ESETBUTTON= CreateWindowEx(TEXT("button"), TEXT("RESET"), // Χρειαζεται για αφαιρεση τραγουδιων
PLAYLIST= CreateWindowEx(NULL, "LISTBOX", // Εδω ειναι η λιστα //
PLAY= CreateWindowEx(NULL, "", // Εδω ειναι η λιστα //
NULL, WS_BORDER | WS_CHILD | WS_VISIBLE | ES_AUTOVSCROLL | LBS_NOTIFY,
STOP= CreateWindowEx(NULL, "", // Εδω ειναι η λιστα //

ΚαραλήςΓιώργος

Επεξήγηση γλώσσας C++

NEXT= CreateWindowEx(NULL, "", // Εδω ειναι η λιστα //
PREVIOUS= CreateWindowEx(NULL, "", // Εδω ειναι η λιστα //
Read me= CreateWindowEx(NULL, "LISTBOX", // Διαγραφη //
CALLLISTBOX2= CreateWindowEx(NULL, "LISTBOX",   // Διαγραφη //
MoveToArrayώτα ονόματα μεσα από τις διατάξεις οπότε αυτές μπορούν να λειτουργήσουν εύκολα
int xx = atoi(STRNAMES[0].c_str()); //μετατροπή string μέσα στο int needs: (#include <stdlib.h>)
string temp;//προσπάθεια συμπλήρωσης των λιστών 1
for (i=1; i<=xx; i++){
listBoxStr[i]=STRNAMES[i];
SendMessage(NAMELISTBOX, LB_ADDSTRING, 0, (LPARAM) listBoxStr[i]);
} 1
break; με το break το πρόγραμμα σταματάει
ofNames.open("Data/Names.txt", ios::app);//ios::app βοηθαει στην επεξεργασία του txt file, οπότε η προηγούμενη είσοδος δεν αλλάχτηκε η διαγράφτηκε. 2
ofRead me.open("Data/Read me.txt", ios::app);
ofRed.open("Data/Red.txt", ios::app);
ofReportBox.open("Data/ReportBox.txt", ios::app); 2x
//checking for errors
if (ofNames.fail()) 3
{
cerr <<"Error opening file Names" <<endl;
exit(1);
}
if(ofRead me.fail()){
cerr <<"Error opening file Read me" <<endl;
exxit(1);
}
if(of8_2_1981.fail()){
cerr <<"Error opening file 8_2_1981" <<endl;
exit(1);
}
if(ofReportBox.fail()){
cerr <<"Error opening file ReportBox" <<endl;
exit(1); 3
}
//πέρνοντας το αρχείο από το NAMEBOX και το αποθηκεύει στό char SavedText[] με μέγιστο μήκος κειμένου 200 και τελικά αποθηκεύει το έγραφο στον φάκελο.
WriteData(NAMEBOX,"ofNames");   4
WriteData(Pikatsu, "ofRead me");
WriteData(Red, "ofRed");
writeData(REPORTBOX, "ofReportBox");
return 0;
} 4
//----------if button "RETRIEVE" is pressed----------
if (LOWORD(wParam) == 2) { 4// Ξεκιναει την διαδικασια αναζητησης απο τα τραγουδια που εχω ριξει στο φακελο με το προγραμμα //
if (GetWindowTextLength(NAMEBOX) != 0){ // if the box is not empty, open Array and start searching
ifNames.open("Data/Names.txt");

ΚαραλήςΓιώργος

```
STRLen= NumOfLines;
SpaceCheckRemove(NAMEBOX, "ifNames", true); i=1;//fixing the data to be searched //fixed data
is in FixedData[] and searching the data in the array STRNAMES
while(i<=STRLen){
if(FixedData==STRNAMES[i]){
cout<< STRNAMES[i]<<"       ";
ifRead me.open("Data/Read me.txt");
ifRed.open("Data/Red.txt");
ifReportBox.open("Data/ReportBox.txt");
for (int j=1; j<=i; j++)
getline(ifRead me,SearchCall1);
getline(ifRed,SearchCall2);
getline(ifReportBox,SearchReport);
}
ifRead me.close();
cout<<SearchCall1<<"       "<< SearchCall2<<"         "<<SearchReport<<endl;
}                                                    i++;
i=0;
}
ifNames.close();
ifRead me.close();
ifRed.close();
ifReportBox.close(); 4
}
//-----------------------------------------if button "RESET" is pressed-----------------------------------------
------
if(LOWORD(wParam)==3){ 5
cout<<"Are you sure you want to reset? (reply with 'y' for yes or 'n' for no) and press
enter"<<endl;
cin>>c;
if(c=='y'||c=='Y'|| c=='υ'||c=='Y'){
ofNames.open("Data/Names.txt", ofstream::out | ofstream::trunc);
ofRead me.open("Data/Read me.txt",ofstream::out | ofstream::trunc);
ofRed.open("Data/Red.txt", ofstream::out | ofstream::trunc);
ofReportBox.open("Data/ReportBox.txt", ofstream::out | ofstream::trunc);
ofNames.close();
ofRead me.close();
ofRed.close();
ofReportBox.close();
cout<< "all the files have been deleted now, programme will restart"<<endl;
exit(1);
}
else if (c=='n'||c=='N'|| c=='v'||c=='N'){
cout<<"nothing was deleted, you may close this window"<<endl;
}
}
break;}
/* Upon destruction, tell the main thread to stop */
case WM_DESTROY: {
PostQuitMessage(0);
break;
```

```
}
\ll other messages (a lot of them) are processed using default procedures */
default:
return DefWindowProc(hwnd, Message, wParam, lParam);
}
return 0;
}
/* The 'main' function of Win32 GUI programs: this is where execution starts */
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int
nCmdShow) {
WNDCLASSEX wc; /* A properties struct of our window */
HWND hwnd; /* A 'HANDLE', hence the H, or a pointer to our window */
MSG msg; /* A temporary location for all messages */
/* zero out the struct and set the stuff we want to modify */
memset(&wc,0,sizeof(wc));
wc.cbSize = sizeof(WNDCLASSEX);
wc.lpfnWndProc = WndProc; /* T/* his is where we will send messages to */
wc.hInstance = hInstance;
wc.hCursor = LoadCursor(NULL, IDC_ARROW);
White, COLOR_WINDOW is just a #define for a system color, try Ctrl+Clicking it */
wc.hbrBackground = GetSysColorBrush(COLOR_3DFACE);
wc.lpszClassName = "WindowClass";
wc.hIcon = LoadIcon(NULL, IDI_APPLICATION); /* Load a standard icon */
wc.hIconSm = LoadIcon(NULL, IDI_APPLICATION); /* use the name "A" to use the project
icon */
if(!RegisterClassEx(&wc)) {
MessageBox(NULL, "Window Registration Failed!","Error!",MB_ICONEXCLAMATION|MB_OK);
return 0;
}
hwnd =
CreateWindowEx(WS_EX_CLIENTEDGE,"WindowClass","RMA",WS_VISIBLE|WS_OVERLAPPEDWI
NDOW,
CW_USEDEFAULT, /* x */
CW_USEDEFAULT, /* y */
640, /* width */
480, /* height */
NULL,NULL,hInstance,NULL);
if(hwnd == NULL) {
MessageBox(NULL, "Window Creation Failed!","Error!",MB_ICONEXCLAMATION|MB_OK);
return 0;
}
/*
sent to WndProc. Note that GetMessage blocks code flow until it receives something, so
this loop will not produce unreasonably high CPU usage
*/
while(GetMessage(&msg, NULL, 0, 0) > 0) { /* If no error is received... */
TranslateMessage(&msg); /* Translate key codes to chars if present */
DispatchMessage(&msg); /* Send it to WndProc */
}
{ 5
\\ Παιξημο τραγουδιου , απο την  σειρα 516-523
```

ΚαραλήςΓιώργος

```
PlaySound(TEXT ("antegeia.wav"),NULL,SND_SYNC); // Here i define the song
// NULL avoid memory problems and works on the aplycasion.
//SYNC_ is calling to sychroniozed the option flow with the control system flow.
System("pause");
return 0; 5
}
printf( "Διαβασε με\n" ); // Διαβασε με 6
\\Απο την σειρα 528-552 φτιαφημο φακελου
{
double d = 3.14;
ofstream(Songs, std::ios::binary7) .write(reinterpret_cast<char*>(&d), sizeof d);
<< 123 << "abc";
// open file for reading
std::ifstream istrm(filename, std::ios::binary);
if (!istrm.is_open()) {
std::cout << "failed to open " << filename << '\n';
else {
double d;
istrm.read(reinterpret_cast<char*>(&d), sizeof d); // binary input
Int n;
std::string s;
if (istrm >> n >> s)                    // text input
std::cout << "read back from file: " << d << ' ' << n << ' ' << s << '\n'; 6
// Δεν θα βαλω return 0; //Γιατι το
}
printf ("Press ENTER to continue.\n");
getchar (); // wait for input
>
{
initwindow(688.388,"insert image.i.");
readimagefile("g7.jpg",0,0,600,388);
getche();
return 0;
}
return msg.wParam;
}
```

24).DLL.

Η βιβλιοθήκη δυναμικής σύνδεσης (ή DLL) είναι η εφαρμογή της ιδέας κοινής βιβλιοθήκης της Microsoft στα λειτουργικά συστήματα Microsoft Windows και "OS, το DRV (για προγράμματα οδήγησης παλαιού τύπου). Οι μορφές αρχείων για DLL είναι οι ίδιες όπως για τα αρχεία των Windows EXE - δηλαδή το Portable Executable (PE) για Windows 32 bit και 64 bit και το νέο εκτελέσιμο (NE) για Windows 16-bit. Όπως συμβαίνει με τα EXE, τα DLL μπορούν να περιέχουν κώδικα, δεδομένα και πόρους, σε οποιονδήποτε συνδυασμό.
Αρχεία δεδομένων με την ίδια μορφή αρχείου με DLL, αλλά με διαφορετικές επεκτάσεις αρχείων και ενδεχομένως με μόνο τμήματα πόρων, μπορούν να ονομάζονται DLL πόρων. Παραδείγματα τέτοιων αρχείων DLL περιλαμβάνουν βιβλιοθήκες εικονιδίων, μερικές φορές έχουν την επέκταση ICL και αρχεία γραμματοσειρών, με τις επεκτάσεις FON και FOT.
Ένα λειτουργικό σύστημα (OS) είναι λογισμικό συστήματος που διαχειρίζεται πόρους υλικού και λογισμικού υπολογιστών και παρέχει κοινές υπηρεσίες για προγράμματα υπολογιστών.

ΚαραλήςΓιώργος

Επεξήγηση γλώσσας C++

Τα λειτουργικά συστήματα χρονομεριστικής κατανομής απαιτούν αποτελεσματική χρήση του συστήματος και μπορεί επίσης να περιλαμβάνουν λογισμικό λογιστικής για την κατανομή του χρόνου επεξεργαστή, τη μαζική αποθήκευση και την εκτύπωση.
Για λειτουργίες υλικού όπως είσοδο και έξοδο και κατανομή μνήμης, το λειτουργικό σύστημα λειτουργεί ως ενδιάμεσος μεταξύ των προγραμμάτων και του υλικού του υπολογιστή, [1] [2] αν και ο κώδικας εφαρμογής εκτελείται συνήθως απευθείας από το υλικό και συχνά κάνει κλήσεις συστήματος σε OS ή διακόπτεται από αυτήν. Τα λειτουργικά συστήματα βρίσκονται σε πολλές συσκευές που περιέχουν έναν υπολογιστή - από κινητά τηλέφωνα και κονσόλες βιντεοπαιχνιδιών έως διακομιστές ιστού και υπερυπολογιστές.
Το κυρίαρχο λειτουργικό σύστημα επιτραπέζιων υπολογιστών είναι τα Microsoft Windows με μερίδιο αγοράς περίπου 82,74%. Το MacOS της Apple Inc. βρίσκεται στη δεύτερη θέση (13,23%), ενώ οι ποικιλίες του Linux είναι συλλογικά στην τρίτη θέση (1,57%). [3] Στον τομέα των κινητών τηλεφώνων (συνδυασμός smartphone και tablet), το 2017 χρησιμοποιεί μέχρι και το 70% του Google Android [4] και σύμφωνα με τα στοιχεία του τρίτου τριμήνου του 2016, το Android για τα smartphones κυριαρχεί με 87,5% και ρυθμό ανάπτυξης 10,3% ακολουθούμενη από το iOS της Apple με 12,1% και μείωση του μεριδίου αγοράς κατά 5,2% ετησίως, ενώ άλλα λειτουργικά συστήματα ανέρχονται μόλις στο 0,3% [5]. Οι διανομές του Linux κυριαρχούν στους τομείς διακομιστών και υπερυπολογιστών. Άλλες εξειδικευμένες κατηγορίες λειτουργικών συστημάτων, όπως ενσωματωμένα συστήματα και συστήματα σε πραγματικό χρόνο, υπάρχουν για πολλές εφαρμογές.

25).Μεταγλωττιστές

Ο πρώτος μεταγλωττιστής C με κλάσεις ονομάστηκε Cfront, ο οποίος προέρχεται από έναν μεταγλωττιστή C που ονομάζεται CPre. Πρόκειται για ένα πρόγραμμα που έχει σχεδιαστεί για να μεταφράσει τον κώδικα C με τους κλάδους στο συνηθισμένο C. Ένα ενδιαφέρον σημείο που αξίζει να σημειωθεί είναι ότι η Cfront γράφτηκε ως επί το πλείστον σε C με μαθήματα, καθιστώντας τον έναν αυτό-φιλοξενούμενο μεταγλωττιστή (έναν μεταγλωττιστή που μπορεί να καταρτίσει τον εαυτό του). Το Cfront αργότερα εγκαταλείφθηκε το 1993, αφού έγινε δύσκολη η ενσωμάτωση νέων χαρακτηριστικών σε αυτό, δηλαδή οι εξαιρέσεις C ++. Παρ 'όλα αυτά, η Cfront είχε τεράστιο αντίκτυπο στις υλοποιήσεις των μελλοντικών μεταγλωττιστών και στο λειτουργικό σύστημα Unix.
Τα βασικά εργαλεία που χρειάζονται για να ακολουθήσουν αυτά τα μαθήματα είναι ένας υπολογιστής και μια εργαλειομηχανή μεταγλωττιστή που είναι σε θέση να καταρτίσει κώδικα C ++ και να δημιουργήσει τα προγράμματα για να τρέξει σε αυτό.
Το C ++ είναι μια γλώσσα που έχει εξελιχθεί πολύ με την πάροδο των ετών, και αυτά τα μαθήματα εξηγούν πολλά χαρακτηριστικά που προστέθηκαν πρόσφατα στη γλώσσα. Επομένως, για να ακολουθήσετε σωστά τα μαθήματα, απαιτείται ένας πρόσφατος μεταγλωττιστής.
Υποστηρίζει τα χαρακτηριστικά που εισάγονται από το πρότυπο του 2011.
Πολλοί προμηθευτές μεταγλωττιστών υποστηρίζουν τα νέα χαρακτηριστικά σε διαφορετικούς βαθμούς.

26).Τι είναι ένας μεταγλωττιστής;

Οι υπολογιστές κατανοούν μόνο μία γλώσσα και αυτή η γλώσσα αποτελείται από σύνολα οδηγιών που γίνονται από αυτά και τα μηδενικά. Αυτή η γλώσσα υπολογιστή ονομάζεται κατάλληλα γλώσσα μηχανής.
Μια ενιαία οδηγία σε έναν υπολογιστή θα μπορούσε να μοιάζει με αυτό:
00000 10011110

Επεξήγηση γλώσσας C++

Ένα πρόγραμμα γλώσσας μηχανής ενός συγκεκριμένου υπολογιστή που επιτρέπει σε έναν χρήστη να εισάγει δύο αριθμούς, προσθέτει τους δύο αριθμούς μαζί και εμφανίζει το σύνολο μπορεί να περιλαμβάνει τις εξής οδηγίες κώδικα μηχανής:

00000 10011110
00001 11110100
00010 10011110
00011 11010100
00100 10111111
00101 00000000

Όπως μπορείτε να φανταστείτε, ο προγραμματισμός ενός υπολογιστή απευθείας στη γλώσσα του μηχανήματος χρησιμοποιώντας μόνο αυτούς και τα μηδενικά είναι πολύ κουραστικό και επιρρεπές στο σφάλμα. Για να διευκολυνθεί ο προγραμματισμός, αναπτύχθηκαν γλώσσες υψηλού επιπέδου. Τα προγράμματα υψηλού επιπέδου διευκολύνουν επίσης τους προγραμματιστές να επιθεωρούν και να κατανοούν καλύτερα τα προγράμματα του άλλου.
Τώρα που καλύπτεται όλη η απαραίτητη θεωρία, είναι τώρα δυνατό να εξηγηθεί τι πρέπει να προσφέρει η C ++ ως γλώσσα προγραμματισμού. C ++ ...

... είναι μια ανοιχτή γλώσσα τυποποίησης ISO.

Για κάποιο χρονικό διάστημα, η C ++ δεν είχε επίσημο πρότυπο και διατηρήθηκε με ένα de facto πρότυπο, ωστόσο από το 1998 η C ++ τυποποιείται από επιτροπή του ISO.

... είναι μια μεταγλωττισμένη γλώσσα.

Το C ++ μεταγλωττίζεται απευθείας στον κώδικα ενός μηχανήματος, επιτρέποντάς του να είναι μία από τις γρηγορότερες γλώσσες στον κόσμο, αν είναι βελτιστοποιημένη.

... είναι μια γλώσσα που δεν είναι ασφαλής.

Το C ++ είναι μια γλώσσα που αναμένει από τον προγραμματιστή να ξέρει τι κάνει, αλλά επιτρέπει απρόσκοπτα αποτελέσματα ως αποτέλεσμα.

... υποστηρίζει και την εκδήλωση και την εξαγωγή συμπερασμάτων.

Από το πιο πρόσφατο πρότυπο C ++, το C ++ υποστηρίζει τόσο την προφανή όσο και την υποτιθέμενη πληκτρολόγηση, επιτρέποντας την ευελιξία και ένα μέσο αποφυγής περιληπτικών λέξεων όπου είναι επιθυμητό.

... υποστηρίζει και τον έλεγχο στατικού και δυναμικού τύπου.

Το C ++ επιτρέπει να ελέγχονται οι μετατροπές τύπου είτε κατά την κατάρτιση είτε κατά τη διάρκεια εκτέλεσης, προσφέροντας και πάλι έναν άλλο βαθμό ευελιξίας. Ο περισσότερος έλεγχος τύπου C ++ είναι, ωστόσο, στατικός.

... προσφέρει πολλές επιλογές παραδειγματισμού.

Το C ++ προσφέρει αξιοσημείωτη υποστήριξη για παραδειγματικά προγραμματιστικά, γενικά και αντικειμενοστρεφή παραδείγματα προγραμματισμού, καθώς και πολλά άλλα παραδείγματα.
Ως μία από τις πιο συχνά χρησιμοποιούμενες γλώσσες στον κόσμο και ως ανοιχτή γλώσσα, η C ++ έχει ένα ευρύ φάσμα μεταγλωττιστών που τρέχουν σε πολλές διαφορετικές πλατφόρμες που την υποστηρίζουν. Ο κώδικας που χρησιμοποιεί αποκλειστικά τη συνηθισμένη βιβλιοθήκη του C ++ θα εκτελεστεί σε πολλές πλατφόρμες με ελάχιστες ή και καθόλου αλλαγές.

... είναι συμβατή με την C

Η C ++, που είναι μια γλώσσα που δημιουργεί άμεσα C, είναι συμβατή με σχεδόν όλους τους κώδικες C. Η C ++ μπορεί να χρησιμοποιήσει βιβλιοθήκες C με λίγες ή καθόλου τροποποιήσεις του κώδικα των βιβλιοθηκών.

... έχει απίστευτη υποστήριξη βιβλιοθήκης.

Μια αναζήτηση για "βιβλιοθήκη" στον δημοφιλή ιστοχώρο διαχείρισης έργου SourceForge θα αποδώσει περισσότερα από 3000 αποτελέσματα για τις βιβλιοθήκες C

27).Τι είναι το ANSI-C ++;

ΚαραλήςΓιώργος

Επεξήγηση γλώσσας C++

Το ANSI-C ++ είναι το όνομα με το οποίο είναι γνωστό το διεθνές πρότυπο ANSI / ISO για τη γλώσσα C ++. Αλλά πριν δημοσιευθεί αυτό το πρότυπο, η C ++ ήταν ήδη ευρέως διαδεδομένη και επομένως υπάρχει πολύς κώδικας εκεί έξω γραμμένος σε πρότυπο C ++. Αναφερόμενος στο ANSI-C ++, το διαφοροποιεί σαφώς από τον πρότυπο κώδικα C ++, ο οποίος είναι ασύμβατος με μερικούς τρόπους.

Το πρότυπο δημοσιεύθηκε το 1998, ακολουθούμενη από αναθεώρηση το 2003. Ορισμένοι μεταγλωτιστές παλαιότεροι από το πρότυπο εφαρμόζουν ήδη μερικά από τα χαρακτηριστικά του και πολλοί νεότεροι μεταγλωττιστές δεν εφαρμόζουν όλες τις λειτουργίες ANSI-C ++. Εάν έχετε αμφιβολίες για το αν ο μεταγλωττιστής σας θα μπορέσει να καταρτίσει κώδικα ANSI-C ++, μπορείτε να δοκιμάσετε να συντάξετε ένα κομμάτι κώδικα με κάποια από τα νέα χαρακτηριστικά που εισήχθησαν κυρίως μετά τη δημοσίευση του προτύπου. Για παράδειγμα, το ακόλουθο κομμάτι κώδικα χρησιμοποιεί τον τύπο bool και χρησιμοποιεί χώρους ονομάτων και πρότυπα.

Αυτό είναι ένα μέρος του κώδικα γραμμένο σε C ++ που επιτυγχάνει τον ίδιο ακριβώς σκοπό:

int a, b, άθροισμα.

cin >> a;

cin >> b;

άθροισμα = α + β.

cout << άθροισμα << endl;

Ακόμα κι αν δεν μπορείτε να καταλάβετε τον παραπάνω κώδικα, θα πρέπει να είστε σε θέση να εκτιμήσετε πόσο πιο εύκολο θα είναι να προγραμματίσετε στη γλώσσα C ++ σε αντίθεση με τη γλώσσα μηχανής.

Επειδή ένας υπολογιστής μπορεί να κατανοήσει μόνο τη γλώσσα της μηχανής και οι άνθρωποι επιθυμούν να γράψουν σε γλώσσες υψηλού επιπέδου, οι γλώσσες υψηλού επιπέδου πρέπει να ξαναγραφούν (μεταφρασθούν) στη γλώσσα μηχανής σε κάποιο σημείο. Αυτό γίνεται με ειδικά προγράμματα που ονομάζονται μεταγλωττιστές, διερμηνείς ή συναρμολογητές που ενσωματώνονται στις διάφορες εφαρμογές προγραμματισμού.

Το C ++ σχεδιάζεται να είναι μια σύνθετη γλώσσα, που σημαίνει ότι γενικά μεταφράζεται σε γλώσσα μηχανής που μπορεί να γίνει κατανοητή απευθείας από το σύστημα, κάνοντας το παραγόμενο πρόγραμμα εξαιρετικά αποτελεσματικό. Γι 'αυτό χρειάζεται ένα σύνολο εργαλείων, γνωστό ως εργαλείο ανάπτυξης, του οποίου ο πυρήνας είναι ένας μεταγλωττιστής και ο σύνδεσμός του.

28).Προγράμματα κονσόλας

Τα προγράμματα κονσόλας είναι προγράμματα που χρησιμοποιούν κείμενο για επικοινωνία με τον χρήστη και το περιβάλλον, όπως η εκτύπωση κειμένου στην οθόνη ή η ανάγνωση εισόδου από ένα πληκτρολόγιο.

Τα προγράμματα κονσόλας είναι εύκολο να αλληλεπιδράσουν και, γενικά, έχουν μια προβλέψιμη συμπεριφορά που είναι ίδια σε όλες τις πλατφόρμες. Είναι επίσης απλό να εφαρμοστούν και έτσι είναι πολύ χρήσιμο να μάθουν τα βασικά μιας γλώσσας προγραμματισμού: Τα παραδείγματα σε αυτά τα tutorials είναι όλα τα προγράμματα κονσόλας. Ο τρόπος για την γραφή προγραμμάτων κονσόλας εξαρτάται από το συγκεκριμένο εργαλείο που χρησιμοποιείτε.

Ο ευκολότερος τρόπος για τους αρχάριους για τη σύνταξη προγραμμάτων C ++ είναι η χρήση ενός ολοκληρωμένου περιβάλλοντος ανάπτυξης (IDE). Ένα IDE γενικά ενσωματώνει διάφορα εργαλεία ανάπτυξης, συμπεριλαμβανομένου ενός επεξεργαστή κειμένου και εργαλεία για την εγραφή προγραμμάτων απευθείας από αυτό.

IDE	Platform	Console programs
Code::blocks	Windows/Linux/MacOS	Compile console programs using Code::blocks
Visual Studio Express	Windows	Compile console programs using VS Express 2013
Dev-C++	Windows	Compile console programs using Dev-C++

ΚαραλήςΓιώργος

Επεξήγηση γλώσσας C++

If you happen to have a Linux or Mac environment with development features, you should be able to compile any of the examples directly from a terminal just by including C++11 flags in the command for the compiler:

Compiler	Platform	Command
GCC	Linux, among others...	g++ -std=c++0x example.cpp -o example_program
Clang	OS X, among others...	clang++ -std=c++11 -stdlib=libc++ example.cpp -o example_program.

29).Αρχικοποίηση μεταβλητών.

Ο καλύτερος τρόπος για να μάθετε μια γλώσσα προγραμματισμού είναι η εγγραφή προγραμμάτων. Συνήθως, το πρώτο πρόγραμμα για αρχάριους γράφει είναι ένα πρόγραμμα που ονομάζεται "Hello World", το οποίο απλά τυπώνει "Hello World" στην οθόνη του υπολογιστή σας. Αν και είναι πολύ απλή, περιέχει όλα τα βασικά συστατικά του C + + προγράμματα:

```
1-// το πρώτο μου πρόγραμμα στο C ++
#include <iostream>
int main ()
{
std :: cout << "Hello World!";
}}
Hello World!
```

Το πλαίσιο παραπάνω δείχνει τον κώδικα C ++ για αυτό το πρόγραμμα. Ο δεξιός πίνακας εμφανίζει το αποτέλεσμα όταν το πρόγραμμα εκτελείται από έναν υπολογιστή. Οι γκρι αριθμοί στα αριστερά των πλαισίων είναι αριθμοί γραμμής για να διευκολύνουν τη συζήτηση προγραμμάτων και την αναζήτηση σφαλμάτων. Δεν αποτελούν μέρος του προγράμματος.

Ας εξετάσουμε αυτή τη γραμμή προγράμματος κατά γραμμή:

Γραμμή 1: // το πρώτο μου πρόγραμμα στην C ++

Δύο σημάδια κάθετης υποδεικνύουν ότι η υπόλοιπη γραμμή είναι ένα σχόλιο που εισήγαγε ο προγραμματιστής αλλά δεν έχει καμία επίδραση στη συμπεριφορά του προγράμματος. Οι προγραμματιστές τις χρησιμοποιούν για να περιλάβουν σύντομες εξηγήσεις ή παρατηρήσεις σχετικά με τον κώδικα ή το πρόγραμμα. Στην περίπτωση αυτή, πρόκειται για μια σύντομη εισαγωγική περιγραφή του προγράμματος.

Γραμμή 2: #include <iostream>

Οι γραμμές που αρχίζουν με ένα σημάδι κατακερματισμού (#) είναι οδηγίες που διαβάζονται και ερμηνεύονται από αυτό που είναι γνωστό ως preprocessor(προεπεξεργαστής). Πρόκειται για ειδικές γραμμές που ερμηνεύονται πριν αρχίσει η σύνταξη του ίδιου του προγράμματος. Σε αυτή την περίπτωση, η οδηγία #include <iostream>, δίνει εντολή στον προεπεξεργαστή να συμπεριλάβει ένα τμήμα του τυπικού κώδικα C ++, γνωστού ως header iostream, που επιτρέπει την εκτέλεση τυποποιημένων λειτουργιών εισόδου και εξόδου, όπως η εγγραφή της εξόδου αυτού του προγράμματος (Hello World) στην οθόνη.

Γραμμή 3: Μία κενή γραμμή.

Οι κενές γραμμές δεν έχουν καμία επίδραση σε ένα πρόγραμμα. Απλώς βελτιώνουν την αναγνωσιμότητα του κώδικα.

Γραμμή 4: int main ()

Αυτή η γραμμή ενεργοποιεί τη δήλωση μιας λειτουργίας. Ουσιαστικά, μια συνάρτηση είναι μια ομάδα δηλώσεων κώδικα που δίνεται ένα όνομα: στην περίπτωση αυτή, αυτό δίνει το όνομα στην ομάδα των δηλώσεων κώδικα που ακολουθούν.

Η κύρια λειτουργία είναι μια ειδική λειτουργία σε όλα τα προγράμματα C ++. είναι η λειτουργία που ονομάζεται κατά την εκτέλεση του προγράμματος. Η εκτέλεση όλων των προγραμμάτων C ++ ξεκινάει με την κύρια λειτουργία.

Οι γραμμές 5 και 7: {και}

ΚαραλήςΓιώργος

Επεξήγηση γλώσσας C++

Το ανοιχτό στήριγμα ({) στη γραμμή 5 υποδηλώνει την αρχή του ορισμού της λειτουργίας του κύριου και το στήριγμα κλεισίματος (} στη γραμμή 7 δείχνει το άκρο του. Όλα μεταξύ αυτών ίναι το σώμα της λειτουργίας που καθορίζει τι συμβαίνει όταν καλείται το κύριο. Όλες οι λειτουργίες χρησιμοποιούν τιράντες για να υποδείξουν την αρχή και το τέλος των ορισμών τους.

Γραμμή 6: std :: cout << "Hello World!";

Αυτή η γραμμή είναι μια εντολή C ++. Μια δήλωση είναι μια έκφραση που μπορεί πραγματικά να παράγει κάποιο αποτέλεσμα. Είναι το κρέας ενός προγράμματος, προσδιορίζοντας την πραγματική συμπεριφορά του. Οι δηλώσεις εκτελούνται με την ίδια σειρά που εμφανίζονται στο σώμα μιας λειτουργίας.

Αυτή η δήλωση έχει τρία μέρη: Πρώτον, std :: cout, που προσδιορίζει την τυπική συσκευή εξόδου χαρακτήρων (συνήθως αυτή είναι η οθόνη του υπολογιστή). Δεύτερον, ο χειριστής εισαγωγής (<<), ο οποίος υποδεικνύει ότι ό, τι ακολουθεί εισάγεται στο std :: cout. Τέλος, μια πρόταση μέσα σε εισαγωγικά ("Hello world!") Είναι το περιεχόμενο που εισάγεται στην τυπική έξοδο.

Παρατηρήστε ότι η δήλωση τελειώνει με ένα ερωτηματικό (;). Αυτός ο χαρακτήρας σηματοδοτεί το τέλος της δήλωσης, ακριβώς όπως η περίοδος λήγει μια πρόταση στα αγγλικά. Όλες οι δηλώσεις C ++ πρέπει να τελειώνουν με έναν ερωτηματικό. Ένα από τα πιο κοινά σφάλματα σύνταξης στη C ++ είναι να ξεχάσουμε να τερματίσουμε μια εντολή με ένα ερωτηματικό.

Μπορεί να έχετε παρατηρήσει ότι δεν όλες οι γραμμές αυτού του προγράμματος εκτελούν ενέργειες κατά την εκτέλεση του κώδικα. Υπάρχει μια γραμμή που περιέχει ένα σχόλιο (ξεκινώντας από //). Υπάρχει μια γραμμή με μια οδηγία για τον προεπεξεργαστή (ξεκινώντας με #). Υπάρχει μια γραμμή που ορίζει μια συνάρτηση (στην περίπτωση αυτή, η κύρια λειτουργία). Και, τέλος, μια γραμμή με δηλώσεις που τελειώνουν με ένα ερωτηματικό (η εισαγωγή στο cout), το οποίο ήταν εντός του μπλοκ που οριοθετείται από τις τιράντες/άγκιστρα ({}) της κύριας λειτουργίας.

Το πρόγραμμα έχει διαρθρωθεί σε διαφορετικές γραμμές και έχει κατανεμηθεί σωστά, ώστε να καταστεί ευκολότερο να γίνει κατανοητό για τους ανθρώπους που το διαβάζουν. Αλλά η C ++ δεν έχει αυστηρούς κανόνες για την εσοχή ή για το πώς να χωρίσει τις οδηγίες σε διαφορετικές γραμμές. Για παράδειγμα, αντί για

```
int main ()
{
std::cout << " Hello World!";
}
```

Θα έπρεπε να γράψουμε:

```
int main () { std::cout << "Hello World!"; }
```

όλα σε μια γραμμή, και αυτό θα είχε ακριβώς την ίδια έννοια με τον προηγούμενο κώδικα.

Στην C ++, ο διαχωρισμός μεταξύ των δηλώσεων καθορίζεται με μιά τελεία , με τον διαχωρισμό σε διαφορετικές γραμμές που δεν έχουν σημασία για το σκοπό αυτό. Πολλές δηλώσεις μπορούν να γραφτούν σε μία γραμμή ή κάθε δήλωση μπορεί να είναι στη δική της γραμμή. Η διαίρεση του κώδικα σε διαφορετικές γραμμές χρησιμεύει μόνο για να γίνει πιο γνωστη και σχηματική για τους ανθρώπους που μπορεί να το διαβάσει, αλλά δεν έχει καμία επίδραση στην πραγματική συμπεριφορά του προγράμματος.

Τώρα, προσθέστε μια επιπλέον δήλωση στο πρώτο μας πρόγραμμα:

```
// my second program in C++
#include <iostream>
int main ()
{
std::cout << "Hello World! ";
std::cout << "I'm a C++ program";
}
```

```
Hello World! I'm a C++ program
```

ΚαραλήςΓιώργος

Σε αυτή την περίπτωση, το πρόγραμμα εκτέλεσε δύο εισαγωγές στο std :: cout σε δύο διαφορετικές δηλώσεις. Για άλλη μια φορά, ο διαχωρισμός σε διαφορετικές γραμμές κώδικα δίνει απλώς μεγαλύτερη αναγνωσιμότητα στο πρόγραμμα, δεδομένου ότι το κύριο θα μπορούσε να είναι απόλυτα έγκυρο και να ορίζεται με αυτόν τον τρόπο:

```
int main () { std::cout << " Hello World! "; std::cout C++ program "; }
```

ο πηγαίος κώδικας θα μπορούσε επίσης να χωριστεί σε περισσότερες γραμμές κώδικα αντί:

```
int main ()
{
std::cout <<
"Hello World!";
std::cout
<< "I'm a C++ program";
}
```

Και το αποτέλεσμα θα ήταν και πάλι ακριβώς το ίδιο όπως στα προηγούμενα παραδείγματα.
Οι οδηγίες του προ-επεξεργαστή (εκείνες που αρχίζουν με #) είναι εκτός αυτού του γενικού κανόνα, καθώς δεν είναι δηλώσεις. Είναι γραμμές που διαβάζονται και επεξεργάζονται από τον προεπεξεργαστή πριν ξεκινήσει η σωστή σύνταξη. Οι οδηγίες του προ-επεξεργαστή πρέπει να διευκρινίζονται στη δική τους γραμμή και, επειδή δεν είναι δηλώσεις, δεν χρειάζεται να τελειώνουν με ερωτηματικό (;).
Τα σχόλια δεν επηρεάζουν τη λειτουργία του προγράμματος. Παρέχουν όμως ένα σημαντικό εργαλείο για να τεκμηριώσετε άμεσα μέσα στον πηγαίο κώδικα τι κάνει το πρόγραμμα και πώς λειτουργεί.
Το C ++ υποστηρίζει δύο τρόπους σχολιασμού κώδικα:

```
// line comment
/* block comment */
```

Ο πρώτος από αυτούς, γνωστός ως σχόλιο γραμμής, απορρίπτει τα πάντα από το σημείο όπου βρίσκονται τα σύμβολα (//) μέχρι το τέλος της ίδιας γραμμής. Το δεύτερο, γνωστό ως μπλοκ σχόλιο, απορρίπτει τα πάντα μεταξύ των χαρακτήρων / * και της πρώτης εμφάνισης των * / χαρακτήρων, με τη δυνατότητα να συμπεριληφθούν πολλές γραμμές.
Ας προσθέσουμε σχόλια στο δεύτερο μας πρόγραμμα:

```
/ * my second program C ++
with lots of comends * /
#include <iostream>
int main ()
{
std :: cout << "Hello World!"; // εκτυπώνει Hello World!
std :: cout << "Είμαι ένα πρόγραμμα C ++"? // εκτυπώσεις Είμαι ένα πρόγραμμα C ++
}}
Hello world I’m a programC ++
```

Εάν τα σχόλια συμπεριλαμβάνονται στον πηγαίο κώδικα ενός προγράμματος χωρίς τη χρήση των συνδυασμών χαρακτήρων σχολίων //, / * ή * /, ο μεταγλωττιστής τις παίρνει σαν να ήταν εκφράσεις C ++, πιθανότατα προκαλώντας την αποτυχία της σύνταξης με ένα ή περισσότερα, μηνύματα σφάλματος.
Χρησιμοποιώντας τον χώρο ονομάτων std
Εάν έχετε δει τον κώδικα C ++ πριν, ίσως έχετε δει cout να χρησιμοποιηθεί αντί του std :: cout. Και οι δύο ονομάζουν το ίδιο αντικείμενο: το πρώτο χρησιμοποιεί το ανεπίσημο όνομά του (cout), ενώ το δεύτερο το χαρακτηρίζει απευθείας μέσα στο χώρο ονομάτων std (ως std :: cout). cout είναι μέρος της τυπικής βιβλιοθήκης και όλα τα στοιχεία της τυπικής βιβλιοθήκης C ++ δηλώνονται μέσα σε αυτό που ονομάζεται namespace: ο χώρος ονομάτων std.
Για να αναφερθούμε στα στοιχεία του χώρου ονομάτων std, ένα πρόγραμμα πρέπει είτε να πληροί κάθε χρήση στοιχείων της βιβλιοθήκης, είτε να εισαγάγουμε την ορατότητα των

στοιχείων του. Ο πιο συνηθισμένος τρόπος για την εισαγωγή ορατότητας αυτών των στοιχείων είναι η χρήση δηλώσεων: std
Αύτη η δήλωση επιτρέπει την πρόσβαση σε όλα τα στοιχεία του χώρου ονομάτων std. Έχοντας αυτό υπόψη, το τελευταίο παράδειγμα μπορεί να ξαναγραφεί για να γίνει απρόσκοπτη χρήση του cout ως εξής:

```
// my second program C ++
#include <iostream>
χρησιμοποιώντας τον χώρο ονομάτων std.
int main ()
{
cout << "Hello World!";
cout << "I'm a program  C ++"?
}}
'm a programHello world!  C ++
```

Και οι δύο τρόποι πρόσβασης στα στοιχεία του χώρου ονομάτων std (ρητή πιστοποίηση και χρήση δηλώσεων) ισχύουν στην C ++ και παράγουν την ίδια ακριβώς συμπεριφορά. Για απλότητα και για βελτίωση της αναγνωσιμότητας, τα παραδείγματα σε αυτά τα μαθήματα θα χρησιμοποιούν πιο συχνά αυτή την τελευταία προσέγγιση με τη χρήση του δηλώσεις, αν και σημειώνουμε ότι η ρητή πιστοποίηση είναι ο μόνος τρόπος για να διασφαλιστεί ότι οι συγκρούσεις ονόματος δεν θα συμβούν ποτέ.
Με διαφορετικά επίπεδα ακρίβειας, ανάλογα με τον τύπο των τριών κυμαινόμενων σημείων. Τύπος Boolean: Ο τύπος boolean, γνωστός σε C ++ ως bool, μπορεί να αντιπροσωπεύει μόνο μία από τις δύο καταστάσεις, αληθείς ή ψευδείς. Εδώ είναι ο πλήρης κατάλογος των θεμελιωδών τύπων σε C ++: Ομάδες Τύπου Ομάδας * Σημειώσεις για μέγεθος / ακρίβεια Τύποι χαρακτήρων char Ακριβώς ένα byte σε μέγεθος. Τουλάχιστον 8 bits. char16_t Δεν είναι μικρότερο από char. Τουλάχιστον 16 bits. char32_t. Δεν είναι μικρότερο από char16_t. Τουλάχιστον 32 bits. wchar_t Μπορεί να αντιπροσωπεύει το μεγαλύτερο υποστηριζόμενο σετ χαρακτήρων. Ακέραιοι τύποι (υπογεγραμμένοι) υπογεγραμμένοι char Το ίδιο μέγεθος με το char. Τουλάχιστον 8 bits. υπογεγραμμένο σύντομο int Δεν είναι μικρότερο από char. Τουλάχιστον 16 bits. υπογραφή int Δεν είναι μικρότερη από σύντομη. Τουλάχιστον 16 bits. υπογεγραμμένο long int Δεν είναι μικρότερο από int. Τουλάχιστον 32 bit. υπογεγραμμένο μακρύ int int Δεν είναι μικρότερο από το μήκος. Τουλάχιστον 64 bit. Ακέραιοι τύποι (υπογεγραμμένοι) μη υπογεγραμμένοι char (ίδιο μέγεθος με τους υπογεγραμμένους ομολόγους τους) unsigned short int unsigned int unsigned long int unsigned long long int Τύποι πλωτών σημείων float double Ακρίβεια όχι μικρότερη από float long double Ακρίβεια όχι λιγότερο από διπλό Boolean τύπου bool Τύπος άκυρου (nullptr) * Τα ονόματα ορισμένων ακέραιων τύπων μπορούν να συντομευθούν χωρίς τα υπογεγραμμένα και int συστατικά τους - μόνο το μέρος που δεν είναι πλάγια είναι υποχρεωμένο να προσδιορίσει τον τύπο, το μέρος με πλάγιους χαρακτήρες είναι προαιρετικό. Δηλαδή, το υπογεγραμμένο σύντομο int μπορεί να συντομευτεί ως υπογεγραμμένο σύντομο, σύντομο int, ή απλά σύντομο. όλοι προσδιορίζουν τον ίδιο θεμελιώδη τύπο. Μέσα σε κάθε μία από τις παραπάνω ομάδες, η διαφορά μεταξύ των τύπων είναι μόνο το μέγεθός τους (δηλαδή, πόσα καταλαμβάνουν στη μνήμη): ο πρώτος τύπος σε κάθε ομάδα είναι ο μικρότερος και ο τελευταίος είναι ο μεγαλύτερος, με τον κάθε τύπο να είναι τουλάχιστον όπως αυτή που προηγείται της ίδιας ομάδας. Εκτός από αυτό, οι τύποι σε μια ομάδα έχουν τις ίδιες ιδιότητες.Κανένας από τους βασικούς τύπους δεν έχει καθορισμένο κανονικό μέγεθος (αλλά ελάχιστο μέγεθος). Επομένως, ο τύπος δεν απαιτείται (και σε πολλές περιπτώσεις δεν είναι) ακριβώς αυτό το ελάχιστο μέγεθος. Αυτό δεν σημαίνει ότι αυτοί οι τύποι έχουν απροσδιόριστο μέγεθος, αλλά ότι δεν υπάρχει τυποποιημένο μέγεθος σε όλους τους μεταγλωτιστές και τα μηχανήματα. κάθε εφαρμογή μεταγλωττιστή μπορεί να καθορίσει τα μεγέθη για αυτούς τους τύπους που ταιριάζουν καλύτερα στην αρχιτεκτονική όπου θα τρέξει το πρόγραμμα. Αυτή η προδιαγραφή γενικού μεγέθους για τύπους προσφέρει στη γλώσσα C ++ μεγάλη ευελιξία ώστε να προσαρμόζεται ώστε

να λειτουργεί βέλτιστα σε όλα τα είδη πλατφορμών, τόσο παρόν όσο και μέλλον. Τα μεγέθη τύπου παραπάνω εκφράζονται σε bits. τα περισσότερα bits που έχει ένας τύπος, οι πιο διακριτές τιμές που μπορεί να αντιπροσωπεύει, αλλά ταυτόχρονα καταναλώνει επίσης περισσότερο χώρο στη μνήμη.

Για ακέραιους τύπους, οι τιμές που αντιπροσωπεύουν περισσότερο χώρο σημαίνει ότι το εύρος τιμών που μπορούν να αντιπροσωπεύουν είναι μεγαλύτερο.. Για τους τύπους κινητής υποδιαστολής, το μέγεθος επηρεάζει την ακρίβειά τους, έχοντας περισσότερα ή λιγότερα κομμάτια για τα σημαντικά και σημαντικά

Μεταβλητές και τύποι.

Η χρησιμότητα των προγραμμάτων "Hello World" που παρουσιάζονται στο προηγούμενο κεφάλαιο είναι μάλλον αμφισβητήσιμη. Έπρεπε να γράψουμε αρκετές γραμμές κώδικα, να τις συνθέσουμε και στη συνέχεια να εκτελέσουμε τον αντικοίμενο κώδικα, μόνο για να πάρουμε το αποτέλεσμα μιας απλής φράσης γραμμένης στην οθόνη. Σίγουρα θα ήταν πολύ γρηγορότερο να γράψουμε τον εαυτό μας με την πρόταση εξόδου.

Ωστόσο, ο προγραμματισμός δεν περιορίζεται μόνο στην εκτύπωση απλών κειμένων στην οθόνη. Προκειμένου να προχωρήσουμε λίγο περισσότερο και να καταφέρουμε να γράψουμε προγράμματα που εκτελούν χρήσιμα καθήκοντα που πραγματικά μας σώζουν την εργασία, πρέπει να εισαγάγουμε την έννοια των μεταβλητών.

Ας φανταστούμε ότι σας ζητώ να θυμάστε τον αριθμό 5 και στη συνέχεια σας ζητώ να απομνημονεύσετε ταυτόχρονα τον αριθμό 2. Έχετε αποθηκεύσει δύο διαφορετικές τιμές στη μνήμη σας (5 και 2). Τώρα, αν σας ζητήσω να προσθέσετε 1 στον πρώτο αριθμό που είπα, θα πρέπει να διατηρείτε τους αριθμούς 6 (δηλαδή 5 + 1) και 2 στη μνήμη σας. Στη συνέχεια θα μπορούσαμε, για παράδειγμα, να αφαιρέσουμε αυτές τις τιμές και να λάβουμε 4 ως αποτέλεσμα.

Η όλη διαδικασία που περιγράφεται παραπάνω είναι ένα παρόμοιο με αυτό που μπορεί να κάνει ένας υπολογιστής με δύο μεταβλητές. Η ίδια διαδικασία μπορεί να εκφραστεί σε C ++ με το ακόλουθο σύνολο δηλώσεων:

α = 5.

b = 2.

α = α + 1.

result = a - b;

Προφανώς, αυτό είναι ένα πολύ απλό παράδειγμα, δεδομένου ότι έχουμε χρησιμοποιήσει μόνο δύο μικρές ακέραιες τιμές, αλλά θεωρούμε ότι ο υπολογιστής σας μπορεί να αποθηκεύσει εκατομμύρια αριθμούς όπως αυτά ταυτόχρονα και να διεξάγει εξελιγμένες μαθηματικές λειτουργίες μαζί τους.

Τώρα μπορούμε να ορίσουμε τη μεταβλητή ως τμήμα της μνήμης για να αποθηκεύσουμε μια τιμή.

Κάθε μεταβλητή χρειάζεται ένα όνομα που το αναγνωρίζει και το διακρίνει από τα άλλα. Για παράδειγμα, στον προηγούμενο κώδικα τα ονόματα μεταβλητών ήταν a, b και αποτέλεσμα, αλλά θα μπορούσαμε να ονομάζαμε τις μεταβλητές οποιεσδήποτε ονόματα μπορούσαμε να βρούμε, εφ 'όσον ήταν έγκυρα αναγνωριστικά C ++.

Αναγνωριστικά

Ένα έγκυρο αναγνωριστικό είναι μια ακολουθία από ένα ή περισσότερα γράμματα, ψηφία ή χαρακτήρες υπογράμμισης (_). Τα διαστήματα, τα σημεία στίξης και τα σύμβολα δεν μπορούν να αποτελούν μέρος ενός αναγνωριστικού. Επιπλέον, τα αναγνωριστικά πρέπει να αρχίζουν πάντα με ένα γράμμα. Μπορούν επίσης να ξεκινούν με έναν υπογραμμισμένο χαρακτήρα (_), αλλά αυτά τα αναγνωριστικά είναι - στις περισσότερες περιπτώσεις - θεωρούνται δεσμευμένα για λέξεις-κλειδιά που αντιστοιχούν σε compiler ή για εξωτερικά αναγνωριστικά, καθώς και αναγνωριστικά που περιέχουν δύο διαδοχικούς χαρακτήρες υπογράμμισης οπουδήποτε. Σε καμία περίπτωση δεν μπορούν να ξεκινήσουν με ένα ψηφίο.

Επεξήγηση γλώσσας C++

Η C ++ χρησιμοποιεί έναν αριθμό λέξεων-κλειδιών για τον προσδιορισμό των λειτουργιών και των περιγραφών δεδομένων. Επομένως, τα αναγνωριστικά που δημιουργούνται από έναν προγραμματιστή δεν μπορούν να αντιστοιχούν σε αυτές τις λέξεις-κλειδιά. Οι τυπικές δεσμευμένες λέξεις-κλειδιά που δεν μπορούν να χρησιμοποιηθούν για τα αναγνωριστικά που έχουν δημιουργηθεί από προγραμματιστές είναι:

ευθυγράμμιση, ευθυγράμμιση και, και_eq, asm, αυτόματη, bitand, bitor, bool, break, case, catch, char, char16_t, char32_t, class, compl, const, constexpr, διπλό, dynamic_cast, else, enum, explicit, εξαγωγή, εξωτερικό, ψευδές, float, για, φίλος, goto, αν, inline, int, ή_eq, ιδιωτικό, προστατευμένος, δημόσιος, μητρώο, reinterpret_cast, επιστροφή, σύντομη, υπογεγραμμένη, sizeof, στατική, static_assert, static_cast, δομή, διακόπτης, πρότυπο, this, thread_local, throw, true, tryped, typpedef, μη υπογεγραμμένο, χρησιμοποιώντας, εικονικό, άκυρο, πτητικό, wchar_t, ενώ, xor, xor_eq

Οι συγκεκριμένοι μεταγλωττιστές μπορεί επίσης να έχουν επιπλέον συγκεκριμένες αποκλειστικές λέξεις-κλειδιά

Πολύ σημαντική: Η γλώσσα C ++ είναι μια γλώσσα "sensitive-case". Αυτό σημαίνει ότι ένα αναγνωριστικό με κεφαλαία γράμματα δεν είναι ισοδύναμο με ένα άλλο με το ίδιο όνομα αλλά γραμμένο με μικρά γράμματα. Έτσι, για παράδειγμα, η μεταβλητή RESULT δεν είναι η ίδια με τη μεταβλητή αποτελέσματος ή τη μεταβλητή Result. Αυτά είναι τρία διαφορετικά αναγνωριστικά που αναγνωρίζουν τρεις διαφορετικές μεταβλητές.

Οι τιμές των μεταβλητών αποθηκεύονται κάπου σε μη καθορισμένη θέση στη μνήμη του υπολογιστή ως μηδενικά και σε αυτές. Το πρόγραμμά μας δεν χρειάζεται να γνωρίζει την ακριβή τοποθεσία όπου αποθηκεύεται μια μεταβλητή. μπορεί απλώς να αναφέρεται σε αυτό με το όνομά του. Αυτό που πρέπει να γνωρίζει το πρόγραμμα είναι το είδος των δεδομένων που είναι αποθηκευμένα στη μεταβλητή. Δεν είναι το ίδιο για να αποθηκεύσετε έναν απλό ακέραιο όρο, όπως είναι η αποθήκευση ενός γράμματος ή ενός μεγάλου αριθμού κυμαινόμενου σημείου. παρόλο που όλοι αντιπροσωπεύονται με μηδενικά και με αυτά, δεν ερμηνεύονται με τον ίδιο τρόπο και σε πολλές περιπτώσεις δεν καταλαμβάνουν το ίδιο ποσό μνήμης.

Οι βασικοί τύποι δεδομένων είναι βασικοί τύποι που υλοποιούνται απευθείας από τη γλώσσα που αντιπροσωπεύουν τις βασικές μονάδες αποθήκευσης που υποστηρίζονται από τα περισσότερα συστήματα. Μπορούν κυρίως να ταξινομηθούν σε:

Τύποι χαρακτήρων: Μπορούν να αντιπροσωπεύουν ένα μόνο χαρακτήρα, όπως το 'A' ή το '$'. Ο βασικότερος τύπος είναι char, ο οποίος είναι ένα byte χαρακτήρα. Άλλοι τύποι παρέχονται επίσης για ευρύτερους χαρακτήρες.

Αριθμητικοί τύποι ακέραιων αριθμών: Μπορούν να αποθηκεύσουν μια τιμή ολόκληρου αριθμού, όπως 7 ή 1024. Υπάρχουν σε διάφορα μεγέθη και μπορούν είτε να υπογραφούν είτε να μην υπογραφούν, ανάλογα με το αν υποστηρίζουν αρνητικές τιμές ή όχι.

Τύποι κυμαινόμενου σημείου: Μπορούν να αντιπροσωπεύουν άσχημα

Αρχικοποίηση μεταβλητών

Όταν οι μεταβλητές στο παραπάνω παράδειγμα δηλώνονται, έχουν μια απροσδιόριστη τιμή μέχρι να τους δοθεί μια τιμή για πρώτη φορά. Αλλά είναι πιθανό μια μεταβλητή να έχει μια συγκεκριμένη τιμή από τη στιγμή που δηλώνεται. Αυτό ονομάζεται αρχικοποίηση της μεταβλητής.

Στην C ++, υπάρχουν τρεις τρόποι για την προετοιμασία των μεταβλητών. Όλα είναι ισοδύναμα και θυμίζουν την εξέλιξη της γλώσσας τα τελευταία χρόνια:

Το πρώτο, γνωστό ως αρχικοποίηση τύπου c (επειδή κληρονομείται από τη γλώσσα C), αποτελείται από την προσθήκη ενός ίσου σημείου ακολουθούμενου από την τιμή στην οποία αρχίζει η μεταβλητή:

αναγνωριστικό τύπου = αρχική τιμή;

Για παράδειγμα, για να δηλώσουμε μια μεταβλητή τύπου int που ονομάζεται x και να την αρχικοποιήσουμε σε μια τιμή μηδέν από την ίδια στιγμή που δηλώνεται, μπορούμε να γράψουμε:

ΚαραλήςΓιώργος

int x = 0;
Μια δεύτερη μέθοδος, γνωστή ως αρχικοποίηση κατασκευαστή (που εισάγεται από τη γλώσσα C ++), περικλείει την αρχική τιμή μεταξύ των παρενθέσεων (()):
αναγνωριστικό τύπου (αρχική τιμή);
Για παράδειγμα:
int x (0).
Τέλος, μια τρίτη μέθοδος, γνωστή ως ομοιόμορφη αρχικοποίηση, παρόμοια με τα παραπάνω, αλλά με τη χρήση σγουρών ({}) αντί για παρενθέσεις (αυτό εισήχθη με την αναθεώρηση του προτύπου C ++, το 2011):
αναγνωριστικό τύπου {initial_value}.
Για παράδειγμα:
int x {0}.
Και οι τρεις τρόποι αρχικοποίησης των μεταβλητών είναι έγκυροι και ισοδύναμοι στην C ++.
// Initialization of variables
#include <iostream>
χρησιμοποιώντας τον χώρο ονομάτων std.
int main ()
{
int a = 5; // αρχική τιμή: 5
int b (3). // αρχική τιμή: 3
int c {2}. // αρχική τιμή: 2
int αποτέλεσμα? // αρχική τιμή απροσδιόριστη
α = α + β.
αποτέλεσμα = a - c;
cout << αποτέλεσμα;
επιστροφή 0?
}}
Επεξεργασία και εκτέλεση
Πληκτρολογήστε έκπτωση: αυτόματη και decltype
Όταν αρχικοποιηθεί μια νέα μεταβλητή, ο μεταγλωττιστής μπορεί να υπολογίσει ποιος είναι ο τύπος της μεταβλητής αυτόματα από τον αρχικοποιητή. Για αυτό, αρκεί να χρησιμοποιούμε το auto ως τον προσδιοριστή τύπου για τη μεταβλητή:
int foo = 0;
auto bar = foo; // το ίδιο όπως: int bar = foo;
Εδώ, η γραμμή δηλώνεται ως έχει αυτόματο τύπο. Επομένως, ο τύπος της γραμμής είναι ο τύπος της τιμής που χρησιμοποιείται για την αρχικοποίησή της: στην περίπτωση αυτή χρησιμοποιεί τον τύπο του foo, ο οποίος είναι int.
Οι μεταβλητές που δεν έχουν αρχικοποιηθεί μπορούν επίσης να κάνουν χρήση της αφαίρεσης τύπου με τον προσδιοριστή decltype:
int foo = 0;
decltype (foo) // int bar;
Εδώ, η μπάρα δηλώνεται ότι έχει τον ίδιο τύπο με τον foo.
auto και decltype είναι ισχυρά χαρακτηριστικά που προστέθηκαν πρόσφατα στη γλώσσα. Αλλά τα χαρακτηριστικά έκπτωσης τύπου που εισάγουν προορίζονται να χρησιμοποιηθούν είτε όταν ο τύπος δεν μπορεί να ληφθεί με άλλα μέσα είτε όταν το χρησιμοποιεί βελτιώνει την αναγνωσιμότητα του κώδικα. Τα δύο παραπάνω παραδείγματα δεν ήταν πιθανόν ούτε από αυτές τις περιπτώσεις χρήσης. Στην πραγματικότητα, πιθανότατα μειώθηκε η αναγνωσιμότητα, καθώς, κατά την ανάγνωση του κώδικα, κάποιος πρέπει να ψάξει για τον τύπο του foo για να γνωρίζει πραγματικά τον τύπο της ράβδου.

30).Εισαγωγή στις χορδές

Επεξήγηση γλώσσας C++

Οι θεμελιώδεις τύποι αντιπροσωπεύουν τους πιο βασικούς τύπους που χειρίζονται τα μηχανήματα όπου μπορεί να τρέξει ο κώδικας. Αλλά ένα από τα σημαντικότερα πλεονεκτήματα της γλώσσας C ++ είναι το πλούσιο σύνολο των σύνθετων τύπων, των οποίων οι θεμελιώδεις τύποι είναι απλά δομικά στοιχεία.

Ένα παράδειγμα τύπου ένωσης είναι η κλάση συμβολοσειρών. Οι μεταβλητές αυτού του τύπου είναι σε θέση να αποθηκεύουν ακολουθίες χαρακτήρων, όπως λέξεις ή προτάσεις. Ένα πολύ χρήσιμο χαρακτηριστικό!

Μια πρώτη διαφορά με τους θεμελιώδεις τύπους δεδομένων είναι ότι για να δηλωθούν και να χρησιμοποιηθούν αντικείμενα (μεταβλητές) αυτού του τύπου, το πρόγραμμα πρέπει να περιλαμβάνει την κεφαλίδα όπου ορίζεται ο τύπος μέσα στην τυπική βιβλιοθήκη (header <string>):

```
// my first line
#include <iostream>
#include <string>
χρησιμοποιώντας τον χώρο ονομάτων std.
int main ()
{
string mystring;
mystring = "Αυτή είναι μια συμβολοσειρά";
cout << mystring;
επιστροφή 0?
}}
```

Αυτή είναι μια συμβολοσειρά

Επεξεργασία και εκτέλεση

Όπως μπορείτε να δείτε στο προηγούμενο παράδειγμα, οι συμβολοσειρές μπορούν να αρχικοποιηθούν με οποιαδήποτε έγκυρη συμβολοσειρά κυριολεκτικά, ακριβώς όπως οι μεταβλητές του αριθμητικού τύπου μπορούν να αρχικοποιηθούν σε οποιαδήποτε έγκυρη αριθμητική κυριολεκτική. Όπως συμβαίνει με τους θεμελιώδεις τύπους, όλες οι μορφές αρχικοποίησης ισχύουν με συμβολοσειρές:

```
string mystring = "Αυτή είναι μια συμβολοσειρά";
string mystring ("Αυτή είναι μια συμβολοσειρά")?
string mystring {"Αυτή είναι μια συμβολοσειρά"}?
```

Οι χορδές μπορούν επίσης να εκτελέσουν όλες τις άλλες βασικές λειτουργίες που μπορούν να χρησιμοποιήσουν οι βασικοί τύποι δεδομένων, όπως και να δηλώνονται χωρίς αρχική τιμή και να αλλάζουν την αξία τους κατά την εκτέλεση:

```
// πρώτη μου σειρά
#include <iostream>
#include <string>
χρησιμοποιώντας τον χώρο ονομάτων std.
int main ()
{
string mystring;
mystring = "Αυτό είναι το αρχικό περιεχόμενο συμβολοσειρών";
cout << mystring << endl;
mystring = "Αυτό είναι ένα διαφορετικό περιεχόμενο συμβολοσειράς"?
cout << mystring << endl;
επιστροφή 0?
}}
```

Αυτό είναι το αρχικό περιεχόμενο συμβολοσειράς
Αυτό είναι ένα διαφορετικό περιεχόμενο συμβολοσειράς

ΚαραλήςΓιώργος

Επεξήγηση γλώσσας C++

Επεξεργασία και εκτέλεση

Σημείωση: η εισαγωγή του τερματικού χειριστή τερματίζει τη γραμμή (εκτυπώνοντας ένα χαρακτήρα νέας γραμμής και ξεπλύνετε το ρεύμα).

Η κλάση συμβολοσειρών είναι a

Σταθερά.

Τα σταθερά είναι εκφράσεις με σταθερή τιμή.

Λουτρά

Τα κυκλώματα είναι το πιο προφανές είδος σταθερών. Χρησιμοποιούνται για να εκφράζουν συγκεκριμένες τιμές μέσα στον πηγαίο κώδικα ενός προγράμματος. Έχουμε ήδη χρησιμοποιήσει κάποια από τα προηγούμενα κεφάλαια για να δώσουμε συγκεκριμένες τιμές στις μεταβλητές ή για να εκφράσουμε μηνύματα που θέλαμε να εκτυπώσουν τα προγράμματά μας, για παράδειγμα, όταν γράψαμε:

α = 5.

Τα 5 σε αυτό το κομμάτι του κώδικα ήταν μια κυριολεκτική σταθερά.

Οι λογικές σταθερές μπορούν να ταξινομηθούν σε: ακέραιο, κυμαινόμενο σημείο, χαρακτήρες, χορδές, Boolean, δείκτες και κυκλώματα που ορίζονται από το χρήστη.

Αριθμοί ακεραίων

1776

707

-273

Αυτές είναι αριθμητικές σταθερές που προσδιορίζουν ακέραιες τιμές. Παρατηρήστε ότι δεν περιλαμβάνονται σε εισαγωγικά ή οποιοδήποτε άλλο ειδικό χαρακτήρα. είναι μια απλή ακολουθία ψηφίων που αντιπροσωπεύουν έναν ακέραιο αριθμό σε δεκαδική βάση. για παράδειγμα, το 1776 αντιπροσωπεύει πάντα την αξία χίλια επτακόσια εβδομήντα έξι.

Εκτός από τους δεκαδικούς αριθμούς (εκείνοι που οι περισσότεροι από εμάς χρησιμοποιούν καθημερινά), η C ++ επιτρέπει τη χρήση οκταδικών αριθμών (βάση 8) και δεκαεξαδικών αριθμών (βάση 16) ως κυριολεκτικές σταθερές. Για τα οκταδικά κυκλώματα, τα ψηφία προηγούνται με χαρακτήρα 0 (μηδέν). Και για δεκαεξαδικό, προηγούνται οι χαρακτήρες 0x (μηδέν, x). Για παράδειγμα, οι ακόλουθες κυριολεκτικές σταθερές είναι όλες ισοδύναμες μεταξύ τους:

75 // δεκαδικό

0113 // οκταδικό

0x4b // δεκαεξαδικό

Όλα αυτά αντιπροσωπεύουν τον ίδιο αριθμό: 75 (εβδομήντα πέντε), εκφρασμένα ως αριθμός-10, οκταδικό και δεκαεξαδικό, αντίστοιχα.

Αυτές οι κυριολεκτικές σταθερές έχουν έναν τύπο, ακριβώς όπως οι μεταβλητές. Από προεπιλογή, τα integer literals είναι τύπου int. Ωστόσο, ορισμένα κατάληξη μπορούν να επισυναφθούν σε ένα κυριολεκτικό ακέραιο για να καθορίσουν έναν διαφορετικό τύπο ακέραιου αριθμού:

Είδος τροποποίησης τύπου

u ή U δεν έχει υπογραφεί

l ή L μακρά

ll ή LL για μεγάλο χρονικό διάστημα

Το μη υπογεγραμμένο μπορεί να συνδυαστεί με οποιοδήποτε άλλο από τα δύο, με οποιαδήποτε σειρά, για να σχηματίσει ένα μη υπογεγραμμένο μεγάλο ή μη υπογεγραμμένο πολύ μακρύ.

Για παράδειγμα:

75 // int

75u // unsigned int

75 l // μακρύ

75ul // χωρίς υπογραφή

75lu // χωρίς υπογραφή

ΚαραλήςΓιώργος

Σε όλες τις παραπάνω περιπτώσεις, το επίθημα μπορεί να οριστεί χρησιμοποιώντας είτε κεφαλαία είτε πεζά γράμματα.

31).Αριθμοί πλωτών σημείων

Εκφράζουν πραγματικές αξίες, με δεκαδικά ψηφία και / ή εκθέτες. Μπορούν να περιλαμβάνουν είτε ένα δεκαδικό σημείο, έναν χαρακτήρα e (που εκφράζει "δέκα στο Χο ύψος", όπου το X είναι μια ακέραια τιμή που ακολουθεί τον χαρακτήρα e), ή και ένα δεκαδικό και ένα ε χαρακτήρα:

3.14159 // 3.14159
6.02e23 // 6.02 χ 10 ^ 23
1.6e-19 // 1.6 χ 10 ^ -19
3.0 // 3.0

Αυτοί είναι τέσσερις έγκυροι αριθμοί με δεκαδικά ψηφία που εκφράζονται σε C ++. Ο πρώτος αριθμός είναι ο PI, ο δεύτερος είναι ο αριθμός του Avogadro, ο τρίτος είναι το ηλεκτρικό φορτίο ενός ηλεκτρονίου (ένας εξαιρετικά μικρός αριθμός) - όλοι προσεγγίζουν - και ο τελευταίος είναι ο τρίτος αριθμός που εκφράζεται ως floating- σημείο αριθμητική κυριολεκτικά.

Ο προεπιλεγμένος τύπος για τα κυμαινόμενα σημεία γράμματα είναι διπλός. Κυκλώματα κυμαινόμενου σημείου τύπου float ή long double μπορούν να καθοριστούν με την προσθήκη ενός από τα ακόλουθα επιθημάτων:

Τύπος πρόσθεσης
f ή F επιπλέουν
l ή L διπλό

Για παράδειγμα:

14159L // long double
02e23f // float

Οποιοδήποτε από τα γράμματα που μπορούν να ανήκουν σε μια αριθμητική σταθερά με κυμαινόμενο σημείο (e, f, l) μπορεί να γραφτεί είτε με κεφαλαία είτε με κεφαλαία γράμματα χωρίς διαφορά στη σημασία.

Λέξεις χαρακτήρων και συμβολοσειρών

Χαρακτήρες και χαρακτήρες κλώνων περικλείονται σε εισαγωγικά:

'z'
'Π'
"Hello world"
"How are you?"

ι δύο πρώτες εκφράσεις αντιπροσωπεύουν γραμμικούς χαρακτήρες ενός χαρακτήρα και οι ακόλουθοι δύο αντιπροσωπεύουν γράμματα συμβολοσειρών που αποτελούνται από διάφορους χαρακτήρες. Παρατηρήστε ότι για να αναπαριστούμε ένα μόνο χαρακτήρα, το περνάμε μεταξύ μοναδικών εισαγωγικών (') και για να εκφράσουμε μια συμβολοσειρά (η οποία γενικά αποτελείται από περισσότερους από έναν χαρακτήρες), περνάμε τους χαρακτήρες μεταξύ των διπλών εισαγωγικών (").

Τόσο τα μονοκείμενα όσο και τα κορδόνια συμβολοσειρών απαιτούν εισαγωγικά που τις περιβάλλουν για να τα διακρίνουν από πιθανά μεταβλητά αναγνωριστικά ή δεσμευμένες λέξεις-κλειδιά. Παρατηρήστε τη διαφορά μεταξύ αυτών των δύο εκφράσεων:

X
'X'

Εδώ, το x μόνο θα αναφέρεται σε ένα αναγνωριστικό, όπως το όνομα μιας μεταβλητής ή ενός σύνθετου τύπου, ενώ το "x" (που περικλείεται μέσα σε μοναδικά εισαγωγικά) θα αναφέρεται στο χαρακτήρα literal 'x' (ο χαρακτήρας που αντιπροσωπεύει πεζά x γράμμα).

Το γράμμα χαρακτήρων και συμβολοσειρών μπορεί επίσης να αντιπροσωπεύει ειδικούς χαρακτήρες που είναι δύσκολο ή αδύνατο να εκφραστούν διαφορετικά στον πηγαίο κώδικα

ενός προγράμματος, όπως η νέα γραμμή (\ n) ή η καρτέλα (\ t). Αυτοί οι ειδικοί χαρακτήρες είναι όλοι τους προηγούμενοι από έναν χαρακτήρα ανάστροφης κάθετος (\).
Εδώ έχετε μια λίστα με τους κωδικούς escape ενός χαρακτήρα:
Κωδικός διαφυγής Περιγραφή
\ n νέα γραμμή
επιστροφή της μεταφοράς
\ t tab
\ vertical tab
\ b backspace
\ f μορφή τροφής (ροή σελίδας)
\ a ειδοποίηση (μπιπ)
\ 'ένα μοναδικό απόσπασμα (')
\ "διπλό απόσπασμα (")
\? ερωτηματικό (?)
\\ αντίστροφη κάθετος (\)
Για παράδειγμα:
'\ n'
'\ t'
"Αριστερά \ t Δεξιά"
"ένα \ ntwo \ nthree"
Εσωτερικά, οι υπολογιστές αντιπροσωπεύουν χαρακτήρες ως αριθμητικούς κωδικούς: συνηθέστερα χρησιμοποιούν μια επέκταση του συστήματος κωδικοποίησης χαρακτήρων ASCII (ανατρέξτε στον κώδικα ASCII για περισσότερες πληροφορίες). Οι χαρακτήρες μπορούν επίσης να εκπροσωπούνται σε κυριολεκτικά γράμματα χρησιμοποιώντας τον αριθμητικό κωδικό τους, γράφοντας έναν χαρακτήρα ανάστροφης κάθετος (\) ακολουθούμενο από τον κωδικό που εκφράζεται ως οκταδικός (βασικός-8) ή δεκαεξαδικός (βασικός-16) αριθμός. Για μια οκταδική τιμή, η ανάστροφη κάθετο ακολουθείται από τα ψηφία. ενώ για δεκαεξαδικό, ένας χαρακτήρας x εισάγεται μεταξύ του πίσω και του δεκαεξαδικού ψηφίου (για παράδειγμα: \ x20 ή \ x4A).
Αρκετές λέξεις-κλειδιά συμβολοσειρών μπορούν να συνδυαστούν για να σχηματίσουν μια κυριολεκτική λέξη με απλά λόγια διαχωρίζοντάς τα με ένα ή περισσότερα κενά κενά, συμπεριλαμβανομένων καρτελών, νέων γραμμών και άλλων έγκυρων κενών χαρακτήρων. Για παράδειγμα:
"this forms a" "one" "string"
"of the characters"
Τα παραπάνω είναι μια συμβολοσειρά γραμμική ισοδύναμη με:
"this formsa ενιαία character string "
Σημειώστε πώς τα κενά στα εισαγωγικά είναι μέρος του κυριολεκτικού, ενώ εκείνα που βρίσκονται έξω από αυτά δεν είναι.
Ορισμένοι προγραμματιστές χρησιμοποιούν επίσης ένα τέχνασμα για να συμπεριλάβουν λυρικά γράμματα μεγάλου μήκους σε πολλές γραμμές: Στην C ++, μια πίσω κάθετο (\) στο τέλος της γραμμής θεωρείται ένας χαρακτήρας συνέχειας γραμμής που συγχωνεύει τόσο τη γραμμή αυτή όσο και την επόμενη σε μια γραμμή. Επομένως, ο ακόλουθος κώδικας:
x = " string expressed in \
two lines "
is equivalent to:
x = " string that is expressed in two lines "
Όλα τα γράμματα των χαρακτήρων και τα γράμματα των συμβολοσειρών που περιγράφονται παραπάνω είναι κατασκευασμένα από χαρακτήρες τύπου char. Ένας διαφορετικός τύπος χαρακτήρων μπορεί να καθοριστεί χρησιμοποιώντας ένα από τα ακόλουθα προθέματα:
Πρόθεμα Τύπος χαρακτήρων
u char16_t

Επεξήγηση γλώσσας C++

U char32_t
L wchar_t
Σημειώστε ότι, σε αντίθεση με τα επιθέματα τύπου για integer literals, αυτά τα προθέματα είναι case sensitive: πεζά για char16_t και κεφαλαία για char32_t και wchar_t.
Για τα κυκλώματα συμβολοσειρών, εκτός από τα παραπάνω u, U και L, υπάρχουν δύο πρόσθετα προθέματα:
Περιγραφή προθέματος
u8 Η λέξη συμβολοσειράς κωδικοποιείται στο εκτελέσιμο με το UTF-8
R Η λέξη συμβολοσειρά είναι μια ακατέργαστη σειρά
Στις πρώτες χορδές, οι ράβδοι πλάτης και τα μονό και διπλά εισαγωγικά είναι όλοι έγκυροι χαρακτήρες. το περιεχόμενο του κυριολεκτικού στοιχείου ορίζεται από μια αρχική αλληλουχία R "(και μια τελική)", όπου η ακολουθία είναι οποιαδήποτε ακολουθία χαρακτήρων (συμπεριλαμβανομένης μιας κενής ακολουθίας). Το περιεχόμενο της συμβολοσειράς είναι αυτό που βρίσκεται μέσα στην παρένθεση, αγνοώντας την ίδια την οριοθετημένη ακολουθία. Για παράδειγμα:
R "(συμβολοσειρά με \ backslash)"
R "&% $ (συμβολοσειρά με \ backslash) &% $"
Και οι δύο συμβολοσειρές παραπάνω ισοδυναμούν με "string με \\ backslash". Το πρόθεμα R μπορεί να συνδυαστεί με οποιοδήποτε άλλο πρόθεμα, όπως u, L ή u8.
Άλλοι λογοτέχνες
Υπάρχουν τρεις λέξεις-κλειδιά λέξεων-κλειδιών σε C ++: true, false και nullptr:
αλήθεια και ψευδής είναι οι δύο πιθανές τιμές για μεταβλητές τύπου bool.
nullptr είναι η τιμή μηδενικού δείκτη.
bool foo = true;
bool bar = ψευδής?
int * p = nullptr;
πληκτρολόγησαν σταθερές εκφράσεις
Μερικές φορές, είναι απλά βολικό να δώσετε ένα όνομα σε μια σταθερή τιμή:
const διπλό pi = 3,1415926;
const char tab = '\ t';
Στη συνέχεια, μπορούμε να χρησιμοποιήσουμε αυτά τα ονόματα αντί για τα κυριολεκτικά που ορίζονται:
#include <iostream>
χρησιμοποιώντας τον χώρο ονομάτων std.
const διπλό pi = 3,14159;
const char newline = '\ n';
int main ()
{
διπλό r = 5.0; // ακτίνα κύκλου
διπλό κύκλο.
κύκλος = 2 * pi * r;
cout << κύκλος;
cout << newline;
}}
31.4159
Επεξεργασία και εκτέλεση
Οι ορισμοί του προ-επεξεργαστή (#define)
Ένας άλλος μηχανισμός για την ονομασία σταθερών τιμών είναι η χρήση των ορισμών του προεπεξεργαστή. Έχουν την ακόλουθη μορφή:
#define replacement identifier

ΚαραλήςΓιώργος

Μετά από αυτήν την οδηγία, κάθε εμφάνιση αναγνωριστικού στον κώδικα ερμηνεύεται ως αντικατάσταση, όπου η αντικατάσταση είναι οποιαδήποτε ακολουθία χαρακτήρων (μέχρι το τέλος της γραμμής). Αυτή η αντικατάσταση εκτελείται από τον προεπεξεργαστή και συμβαίνει πριν από την κατάρτιση του προγράμματος, προκαλώντας έτσι ένα είδος τυφλής αντικατάστασης: η εγκυρότητα των τύπων ή της σχετικής σύνταξης δεν ελέγχεται με κανέναν τρόπο.

Για παράδειγμα:

#include <iostream>
χρησιμοποιώντας τον χώρο ονομάτων std.
#define PI 3.14159
#define NEWLINE '\ n'
int main ()
{
διπλό r = 5.0; // ακτίνα κύκλου
διπλό κύκλο.
κύκλος = 2 * PI * r;
cout << κύκλος;
cout << NEWLINE;
}}
31.4159

Επεξεργασία και εκτέλεση

Σημειώστε ότι οι #define γραμμές είναι οδηγίες preprocessor, και σαν τέτοιες είναι εντολές μιας γραμμής που -όπως οι δηλώσεις C ++- δεν απαιτούν ερωτηματικά (;) στο τέλος. η οδηγία επεκτείνεται αυτομάτως μέχρι το τέλος της γραμμής. Εάν στην γραμμή βρίσκεται ένα ερωτηματικό, αποτελεί μέρος της ακολουθίας αντικατάστασης και συμπεριλαμβάνεται σε όλα τα αντικατασταθέντα περιστατικά.

33).Λογικοί χειριστές

Μόλις εισαχθούν σε μεταβλητές και σταθερές, μπορούμε να αρχίσουμε να λειτουργούμε μαζί τους χρησιμοποιώντας τους χειριστές. Αυτό που ακολουθεί είναι ένας πλήρης κατάλογος φορέων. Σε αυτό το σημείο, είναι πιθανόν να μην είναι απαραίτητο να γνωρίζετε όλα αυτά, αλλά όλα αναφέρονται εδώ για να χρησιμεύσουν ως αναφορά.

Διαχειριστής εκχώρησης (=)

Ο χειριστής εκχώρησης εκχωρεί μια τιμή σε μια μεταβλητή.

x = 5;

Αυτή η εντολή εκχωρεί την ακέραια τιμή 5 στη μεταβλητή x. Η διαδικασία εκχώρησης πραγματοποιείται πάντα από τα δεξιά προς τα αριστερά και ποτέ το αντίστροφο:

x = y;

Αυτή η δήλωση αποδίδει στη μεταβλητή x την τιμή που περιέχεται στη μεταβλητή y. Η τιμή του x τη στιγμή που εκτελείται αυτή η δήλωση χάνεται και αντικαθίσταται από την τιμή του y. Σκεφτείτε επίσης ότι αναθέτουμε μόνο την τιμή του y στο x κατά τη στιγμή της διαδικασίας εκχώρησης. Επομένως, εάν το y αλλάξει σε μια μεταγενέστερη στιγμή, δεν θα επηρεάσει τη νέα τιμή που λαμβάνεται από το x.

Για παράδειγμα, ας ρίξουμε μια ματιά στον ακόλουθο κώδικα - Έχω συμπεριλάβει την εξέλιξη του περιεχομένου που είναι αποθηκευμένο στις μεταβλητές ως σχόλια:

// operator operator
#include <iostream>
χρησιμοποιώντας τον χώρο ονομάτων std.
int main ()
{

Επεξήγηση γλώσσας C++

int a, b; // α: β, β:;
α = 10; // α: 10, β:;
b = 4; // α: 10, β: 4
a = b; // α: 4, β: 4
b = 7; // α: 4, β: 7
cout << "α:";
cout << a;
cout << "b:";
cout << b;
}}
α: 4 β: 7

Επεξεργασία και εκτέλεση

Το πρόγραμμα αυτό εκτυπώνει στην οθόνη τις τελικές τιμές των a και b (4 και 7, αντίστοιχα). Παρατηρήστε πως δεν επηρεάστηκε από την τελική τροποποίηση του b, αν και δηλώσαμε a = b νωρίτερα.

Οι εργασίες εκχώρησης είναι εκφράσεις που μπορούν να αξιολογηθούν. Αυτό σημαίνει ότι η ίδια η εκχώρηση έχει αξία, και -για θεμελιώδεις τύπους- αυτή η τιμή είναι αυτή που έχει αποδοθεί στη λειτουργία. Για παράδειγμα:

y = 2 + (χ = 5).

Σε αυτή την έκφραση, το y έχει εκχωρηθεί το αποτέλεσμα της προσθήκης 2 και της τιμής μιας άλλης έκθεσης ταξινόμησης (η οποία έχει η ίδια τιμή 5). Είναι σχεδόν ισοδύναμο με:

x = 5;
y = 2 + χ.

Με το τελικό αποτέλεσμα της ανάθεσης 7 στο y.

Η ακόλουθη έκφραση ισχύει επίσης στην C ++:

χ = y = ζ = 5.

Εκχωρεί 5 και στις τρεις μεταβλητές: x, y και z; πάντα από δεξιά προς τα αριστερά.

Αριθμητικοί χειριστές (+, -, *, /,%)

Οι πέντε αριθμητικές πράξεις που υποστηρίζονται από τη C ++ είναι:

περιγραφή του χειριστή
+ προσθήκη
- αφαίρεση
* πολλαπλασιασμός
/ διαίρεση
% modulo

Οι πράξεις προσθήκης, αφαίρεσης, πολλαπλασιασμού και διαίρεσης αντιστοιχούν κυριολεκτικά στους αντίστοιχους μαθηματικούς χειριστές τους. Ο τελευταίος, ο χειριστής modulo, που αντιπροσωπεύεται από ένα ποσοστό% (%), δίνει το υπόλοιπο μιας διαίρεσης δύο τιμών. Για παράδειγμα:

χ = 11% 3.

έχει ως αποτέλεσμα μεταβλητή x που περιέχει την τιμή 2, αφού η διαίρεση 11 με 3 έχει ως αποτέλεσμα 3, με υπόλοιπο 2.

Εκχώρηση ένωσης (+ =, - =, * =, / =,% =, >> =, << =, & =, ^ =, | =)

Οι χειριστές εκχώρησης σύνθετων στοιχείων τροποποιούν την τρέχουσα τιμή μιας μεταβλητής εκτελώντας μια λειτουργία σε αυτήν. Είναι ισοδύναμα με την απόδοση του αποτελέσματος μιας λειτουργίας στον πρώτο τελεστή:

έκφραση ισοδύναμη με ...
y + = χ. y = y + χ.
x = 5; x = χ - 5.
x / y; x = χ / y.
τιμή * = μονάδες + 1; τιμή = τιμή * (μονάδες + 1);

ΚαραλήςΓιώργος

και το ίδιο για όλους τους άλλους φορείς εκχώρησης σύνθεσης. Για παράδειγμα:

```
/ operators operator assignment
#include <iostream>
χρησιμοποιώντας τον χώρο ονομάτων std.
int main ()
{
int a, b = 3,
a = b;
α + = 2. // ισοδύναμο προς a = a + 2
cout << a;
}}
```

Επεξεργασία και εκτέλεση

Αύξηση και μείωση (++, -)

Ορισμένη έκφραση μπορεί να μειωθεί ακόμη περισσότερο: ο χειριστής αύξησης (++) και ο χειριστής μείωσης (-) αυξάνουν ή μειώνουν κατά μία την τιμή που είναι αποθηκευμένη σε μια μεταβλητή. Είναι ισοδύναμα με + = 1 και προς - = 1, αντίστοιχα. Ετσι:

```
++ x;
x + = 1.
x = χ + 1.
```

όλα ισοδυναμούν με τη λειτουργικότητά του. οι τρεις από αυτές αυξάνουν κατά μία τιμή του x. Στους πρώτους μεταγλωττιστές C, οι τρεις προηγούμενες εκφράσεις μπορεί να έχουν παραγάγει διαφορετικό εκτελέσιμο κώδικα ανάλογα με το ποιο από αυτά χρησιμοποιήθηκε. Σήμερα, αυτός ο τύπος βελτιστοποίησης κώδικα εκτελείται γενικά αυτόματα από τον μεταγλωττιστή, οπότε οι τρεις εκφράσεις πρέπει να παράγουν ακριβώς τον ίδιο εκτελέσιμο κώδικα.

Μια ιδιαιτερότητα αυτού του χειριστή είναι ότι μπορεί να χρησιμοποιηθεί τόσο ως πρόθεμα όσο και ως επίθημα. Αυτό σημαίνει ότι μπορεί να γραφτεί είτε πριν από το όνομα της μεταβλητής (++ x) είτε μετά από αυτό (x ++). Αν και σε απλές εκφράσεις όπως x ++ ή ++ x, και οι δύο έχουν ακριβώς το ίδιο νόημα. σε άλλες εκφράσεις στις οποίες αξιολογείται το αποτέλεσμα της διαδικασίας αύξησης ή μείωσης, μπορεί να έχουν σημαντική διαφορά ως προς το νόημά τους: Στην περίπτωση που ο χειριστής αύξησης χρησιμοποιείται ως πρόθεμα (++ x) της τιμής, η έκφραση αξιολογεί μέχρι την τελική τιμή του x, μόλις αυτή έχει ήδη αυξηθεί. Από την άλλη πλευρά, σε περίπτωση που χρησιμοποιείται ως επίθημα (x ++), η τιμή είναι επίσης αυξημένη, αλλά η έκφραση αξιολογείται στην τιμή που είχε το x πριν αυξηθεί. Παρατηρήστε τη διαφορά:

Παράδειγμα 1 Παράδειγμα 2

```
x = 3;
y = ++ χ.
// x περιέχει 4, y περιέχει 4
x = 3;
y = x ++.
// x περιέχει 4, y περιέχει 3
```

Στο παράδειγμα 1, η τιμή που αποδίδεται στο y είναι η τιμή του x μετά την αύξηση. Ενώ στο παράδειγμα 2, η τιμή x είχε πριν αυξηθεί.

Οι χειριστές σχέσης και σύγκρισης (==,! =,>, <,> =, <=)

Δύο εκφράσεις μπορούν να συγκριθούν χρησιμοποιώντας χειριστές σχέσης και ισότητας. Για παράδειγμα, για να μάθουμε αν δύο τιμές είναι ίσες ή αν το ένα είναι μεγαλύτερο από το άλλο. Το αποτέλεσμα μιας τέτοιας ενέργειας είναι είτε αληθές είτε ψευδές (δηλ. Μια τιμή Boolean).

Οι σχεσιακοί χειριστές στη C ++ είναι:

περιγραφή του χειριστή

== Ίσο με

! = Δεν είναι ίση με

<Λιγότερο από

Επεξήγηση γλώσσας C++

> Μεγαλύτερη από
<= Λιγότερο ή ίσο με
> = Μεγαλύτερη ή ίση με
Εδώ υπάρχουν μερικά παραδείγματα:
(7 == 5) // εκτιμάται ως ψευδής
(5> 4) // εκτιμάται ως αληθής
(3! = 2) // εκτιμάται ως αληθής
(6> = 6) // εκτιμάται ως αληθής
(5 <5) // εκτιμάται ως ψευδής
Φυσικά, δεν είναι μόνο αριθμητικές σταθερές που μπορούν να συγκριθούν, αλλά μόνο οποιαδήποτε αξία, συμπεριλαμβανομένων, φυσικά, μεταβλητών. Ας υποθέσουμε ότι a = 2, b = 3 και c = 6, τότε:
(a == 5) // εκτιμάται ως ψευδής, αφού το a δεν είναι ίσο με 5
(a * b> = c) // αξιολογείται ως αληθής, αφού (2 * 3> = 6) είναι αληθές
(b + 4> a * c) // εκτιμάται ως ψευδής, αφού (3 + 4> 2 * 6) είναι ψευδής
((b = 2) == α) // αξιολογείται ως αληθής
Πρόσεχε! Ο χειριστής εκχώρησης (χειριστής =, με ένα ίση σημάδι) δεν είναι ο ίδιος με τον χειριστή σύγκρισης ισότητας (χειριστής ==, με δύο ίσες πινακίδες). η πρώτη (=) εκχωρεί την τιμή στα δεξιά της μεταβλητής στα αριστερά της, ενώ η άλλη (==) συγκρίνει εάν οι τιμές και στις δύο πλευρές του χειριστή είναι ίσες. Επομένως, στην τελευταία έκφραση ((b = 2) == a), εκχωρήσαμε αρχικά την τιμή 2 στο b και κατόπιν τη συγκρίνουμε με a (που αποθηκεύει επίσης την τιμή 2), αποδίδοντας true.
Λογικοί χειριστές (!, &&, ||)
Ο χειριστής ! είναι ο τελεστής C ++ για τη λειτουργία Boolean NOT. Έχει μόνο ένα τελεστή, στα δεξιά του, και το αντιστρέφει, παράγει ψευδή αν το όνομά του είναι αληθινό, και αληθινό αν ο τελεστής είναι ψευδής. Βασικά, επιστρέφει την αντίθετη Boolean τιμή αξιολόγησης του operand. Για παράδειγμα:
!(5 == 5) // εκτιμάται ως ψευδής επειδή η έκφραση στα δεξιά της (5 == 5) είναι αληθής
! (6 <= 4) // εκτιμάται ως αληθής επειδή (6 <= 4) θα ήταν ψευδές
! true // εκτιμάται ως ψευδής
// false // εκτιμάται ως αληθές
Οι λογικοί χειριστές && και || χρησιμοποιούνται όταν αξιολογούνται δύο εκφράσεις για να επιτευχθεί ένα μοναδικό σχεσιακό αποτέλεσμα. Ο χειριστής && αντιστοιχεί στη λογική λειτουργία Boolean AND, η οποία είναι αληθής αν και οι δύο τελεστές είναι αληθινοί και ψευδώς διαφορετικά. Ο ακόλουθος πίνακας δείχνει το αποτέλεσμα του χειριστή && και η αξιολόγηση της έκφρασης a && b:
&& ΦΟΡΕΑΣ (και)
a b a && b
αλήθεια αλήθεια αλήθεια
αληθινό ψεύτικο ψεύτικο
ψευδή αληθινό ψεύτικο
ψευδές ψεύτικο ψεύτικο
Ο χειριστής || αντιστοιχεί στη λογική λειτουργία Boolean Ή, η οποία αποδίδεται αληθής εάν ένας από τους τελεστές του είναι αληθής, οπότε είναι ψευδής μόνο όταν και οι δύο τελεστές είναι ψευδείς. Εδώ είναι τα πιθανά αποτελέσματα ενός || b:
|| ΕΠΙΧΕΙΡΗΣΗ (ή)
a b a || σι
αλήθεια αλήθεια αλήθεια
αληθινό ψεύτικο
ψευδή αλήθεια
ψευδές ψεύτικο ψεύτικο

ΚαραλήςΓιώργος

Για παράδειγμα:
((5 == 5) && (3> 6)) // εκτιμάται σε false (true && false)
((5 == 5) || (3> 6)) // αξιολογείται ως true (true || false)
Όταν χρησιμοποιείτε τους λογικούς χειριστές, η C ++ αξιολογεί μόνο ό, τι είναι απαραίτητο από αριστερά προς τα δεξιά για να βγάλει το συνδυασμένο σχεσιακό αποτέλεσμα, αγνοώντας το υπόλοιπο. Επομένως, στο τελευταίο παράδειγμα ((5 == 5) || (3> 6)), η C ++ αξιολογεί πρώτα εάν το 5 == 5 είναι αληθές και εάν ναι, ποτέ δεν ελέγχει εάν το 3> 6 είναι αληθές ή όχι. Αυτή η διαδικασία είναι γνωστή ως βραχυκύκλωμα και λειτουργεί έτσι για αυτούς τους χειριστές:
βραχυκύκλωμα χειριστή
&& αν η αριστερή πλευρά της έκφρασης είναι ψευδής, το συνδυασμένο αποτέλεσμα είναι ψευδές (η έκφραση της δεξιάς πλευράς δεν αξιολογείται ποτέ).
|| εάν η αριστερής πλευρά έκφραση είναι αληθής, το συνδυασμένο αποτέλεσμα είναι αληθές (η έκφραση της δεξιάς πλευράς δεν αξιολογείται ποτέ).
Αυτό είναι ιδιαίτερα σημαντικό όταν η δεξιά έκφραση έχει παρενέργειες, όπως οι μεταβαλλόμενες τιμές:
εάν ((i <10) && (++ i <n)) {/****} // σημειώστε ότι η συνθήκη αυξάνεται i
Εδώ, η συνδυασμένη έκφραση υπό όρους θα αυξανόταν από ένα, αλλά μόνο αν η κατάσταση στα αριστερά του && είναι αληθής, γιατί διαφορετικά, η συνθήκη στη δεξιά πλευρά (++ i <n) δεν αξιολογείται ποτέ.

34).Υποχρεωτικός τριμερής χειριστής (?)

Ο υποκείμενος χειριστής αξιολογεί μια έκφραση, επιστρέφοντας μία τιμή εάν αυτή η έκφραση αξιολογείται ως αληθής, και μια άλλη εάν η έκφραση αξιολογείται ως ψευδής. Η σύνταξή του είναι:
κατάσταση ? αποτέλεσμα1: αποτέλεσμα2
Εάν η συνθήκη είναι αληθής, ολόκληρη η έκφραση αξιολογείται ως αποτέλεσμα1, και διαφορετικά θα προκύψει2.
7 == 5; 4: 3 // εκτιμάται σε 3, δεδομένου ότι το 7 δεν είναι ίσο με 5.
7 == 5 + 2; 4: 3 // εκτιμάται σε 4, δεδομένου ότι το 7 είναι ίσο με 5 + 2.
5> 3; a: b // εκτιμάται στην τιμή του a, αφού το 5 είναι μεγαλύτερο από 3.
a> b; a: b // αξιολογείται σε όποιο είναι μεγαλύτερο, a ή b.
Για παράδειγμα:
// υπό όρους χειριστής
#include <iostream>
χρησιμοποιώντας τον χώρο ονομάτων std.
int main ()
{
int a, b, γ;
α = 2.
b = 7;
c = (a> b); α: b;
cout << c << '\ n';
}}
Επεξεργασία και εκτέλεση
Σε αυτό το παράδειγμα, το a ήταν 2 και το b ήταν 7, έτσι ώστε η έκφραση που αξιολογείται (a> b) δεν ήταν αληθής, έτσι η πρώτη τιμή που καθορίστηκε μετά το ερωτηματικό απορρίφθηκε υπέρ της δεύτερης τιμής) που ήταν b (με τιμή 7).
Τελεστής γραμμής (,)

Επεξήγηση γλώσσας C++

Ο χειριστής με κόμματα (,) χρησιμοποιείται για να διαχωρίσει δύο ή περισσότερες εκφράσεις που περιλαμβάνονται εκεί όπου αναμένεται μόνο μία έκφραση. Όταν το σύνολο των εκφράσεων πρέπει να αξιολογηθεί για μια τιμή, λαμβάνεται υπόψη μόνο η σωστή έκφραση.
Για παράδειγμα, τον ακόλουθο κώδικα:
α = (β = 3, b + 2).
θα αναθέσει πρώτα την τιμή 3 στο b, και στη συνέχεια θα εκχωρήσει το b + 2 στη μεταβλητή a. Έτσι, στο τέλος, η μεταβλητή a θα περιέχει την τιμή 5 ενώ η μεταβλητή b θα περιέχει τιμή 3.
Δυναμικοί χειριστές (&, |, ^, ~, <<, >>)
Οι δυαδικοί χειριστές τροποποιούν τις μεταβλητές λαμβάνοντας υπόψη τα μοτίβα δυαδικών ψηφίων που αντιπροσωπεύουν τις τιμές που αποθηκεύουν.
περιγραφή ισοδύναμου χειριστή
& ΚΑΙ Bitwise ΚΑΙ
| Ή ΣΗΜΕΙΩΣΗ ΑΛΛΩΝ
^ XOR Σφάλμα αποκλειστικά OR
~ NOT Unary συμπλήρωμα (αναστροφή bit)
<< SHL Shift bits αριστερά
>> SHR Μετακινήστε τα κομμάτια δεξιά

35).Λειτουργία ρητής χύτευσης τύπου

Οι χειριστές τύπου casting επιτρέπουν τη μετατροπή μιας τιμής ενός δεδομένου τύπου σε άλλο τύπο. Υπάρχουν διάφοροι τρόποι για να το κάνετε αυτό στην C ++. Το πιο απλό, το οποίο έχει κληρονομηθεί από τη γλώσσα C, είναι να προηγείται της έκφρασης προς μετατροπή από τον νέο τύπο που περικλείεται μεταξύ των παρενθέσεων (()):
int i;
float f = 3,14.
i = (int) f.
Ο προηγούμενος κώδικας μετατρέπει τον αριθμό κυμαινόμενου σημείου 3.14 σε ακέραια τιμή (3). το υπόλοιπο χάθηκε. Εδώ, ο χειριστής τύπου castcasting ήταν (int). Ένας άλλος τρόπος να κάνουμε το ίδιο πράγμα με τη C ++ είναι να χρησιμοποιήσουμε τη λειτουργική σημειογραφία που προηγείται της έκφρασης που πρέπει να μετατραπεί από τον τύπο και να περικλείει την έκφραση μεταξύ των παρενθέσεων:
i = int (f).
Και οι δύο τρόποι τύπων χύτευσης ισχύουν στην C ++.
μέγεθος του
Αυτός ο χειριστής δέχεται μία παράμετρο, η οποία μπορεί να είναι είτε ένας τύπος είτε μια μεταβλητή και επιστρέφει το μέγεθος σε bytes αυτού του τύπου ή αντικειμένου:
x = μέγεθοςof (char);
Εδώ, η x έχει εκχωρηθεί η τιμή 1, επειδή char είναι ένας τύπος με μέγεθος ενός byte.
Η τιμή που επιστρέφεται από sizeof είναι μια σταθερά χρόνου μεταγλώττισης, οπότε καθορίζεται πάντα πριν από την εκτέλεση του προγράμματος.
Άλλοι χειριστές
Αργότερα σε αυτά τα μαθήματα, θα δούμε μερικούς ακόμα χειριστές, όπως αυτούς που αναφέρονται στους δείκτες ή τις ιδιαιτερότητες του αντικειμενοστρεφούς προγραμματισμού.
Προτεραιότητα των φορέων εκμετάλλευσης
Μία έκφραση μπορεί να έχει πολλαπλούς χειριστές. Για παράδειγμα:
x = 5 + 7% 2.
Στην C ++, η παραπάνω έκφραση εκχωρεί πάντοτε το 6 στη μεταβλητή x, επειδή ο χειριστής% έχει υψηλότερη προτεραιότητα από τον + χειριστή και πάντα αξιολογείται πριν. Τμήματα των εκφράσεων μπορούν να περικλείονται σε παρένθεση για να παρακάμψουν αυτήν την σειρά

προτεραιότητας ή για να καταστήσουν σαφή την επιδιωκόμενη επίδραση. Παρατηρήστε τη διαφορά:

χ = 5 + (7% 2). // x = 6 (ίδιο και χωρίς παρένθεση)

x = (5 + 7)% 2, // x = 0

Από τη μέγιστη έως τη μικρότερη προτεραιότητα, οι χειριστές C ++ αξιολογούνται με την ακόλουθη σειρά:

Ομάδα προτεραιότητας Επίπεδο Περιγραφή χειριστή Ομαδοποίηση

1 Πεδίο εφαρμογής :: προσδιοριστής περιοχής από αριστερά προς δεξιά

2 Postfix (unary) ++ - αύξηση / μείωση postfix Από αριστερά προς δεξιά

() λειτουργικές μορφές

[] δείκτης

. -> πρόσβαση μελών

3 Πρόθεμα (unary) ++ - αύξηση πρόβλεψης / μείωση από δεξιά προς τα αριστερά

~ ~! δυαδική NOT / λογική NOT

+ - ενιαίο πρόθεμα

& * αναφορά / υποχώρηση

νέα διαγραφή κατανομής / ανακατανομής

sizeof πακέτου παραμέτρων

(τύπου) χύτευση τύπου στυλ C

4 Δείκτης-προς-μέλος. * -> * δείκτης πρόσβασης Από αριστερά προς τα δεξιά

5 Αριθμητική: κλιμάκωση * /% πολλαπλασιασμός, διαίρεση, μοντελο Από αριστερά προς δεξιά

6 Αριθμητική: προσθήκη + - προσθήκη, αφαίρεση Από αριστερά προς δεξιά

7 Μετατόπιση με στροφές << >> μετακίνηση αριστερά, μετακίνηση δεξιά Αριστερά προς τα δεξιά

8 Relational <> <=> = χειριστές σύγκρισης από αριστερά προς δεξιά

9 Ισότητα ==! = Ισότητα / ανισότητα Από αριστερά προς τα δεξιά

10 & & bitwise Αριστερά προς τα δεξιά

11 Αποκλειστική ή ^ δυαδική XOR Από αριστερά προς τα δεξιά

12 Συμπεριλαμβανομένων ή bitwise OR από αριστερά προς τα δεξιά

13 Σύζευξη && λογικό AND Από αριστερά προς τα δεξιά

14 Διαζύγιο || λογικό από αριστερά προς δεξιά

15 Εκφάνσεις επιπέδου εκχώρησης = * = / =% = + = - =

>> = << = & = ^ = | = αντιστοιχία εκχώρησης / σύνθεσης από δεξιά προς αριστερά

?: υπό όρους χειριστής

16 Sequencing, διαχωριστικό με κόμμα Από αριστερά προς τα δεξιά

Όταν μια έκφραση έχει δύο χειριστές με το ίδιο επίπεδο προτεραιότητας, η ομαδοποίηση καθορίζει ποιος αξιολογείται πρώτα: είτε από αριστερά προς δεξιά είτε από δεξιά προς τα αριστερά.

Η συμπλήρωση όλων των δευτερευόντων δηλώσεων σε παρένθεση (ακόμη και αυτές που δεν είναι απαραίτητες λόγω της προτεραιότητάς τους) βελτιώνει την αναγνωσιμότητα του κώδικα.

Εκχώρηση ένωσης (+ =, - =, * =, / =,% =, >> =, << =, & =, ^ =, | =)

Οι χειριστές εκχώρησης σύνθετων στοιχείων τροποποιούν την τρέχουσα τιμή μιας μεταβλητής εκτελώντας μια λειτουργία σε αυτήν. Είναι ισοδύναμα με την απόδοση του αποτελέσματος μιας λειτουργίας στον πρώτο τελεστή:

έκφραση ισοδύναμη με ...

y + = χ. y = y + χ.

x = 5; x = χ - 5.

x / y; x = χ / y.

τιμή * = μονάδες + 1; τιμή = τιμή * (μονάδες + 1);

και το ίδιο για όλους τους άλλους φορείς εκχώρησης σύνθεσης. Για παράδειγμα:

// operators operator assignment

Επεξήγηση γλώσσας C++

#include <iostream>
χρησιμοποιώντας τον χώρο ονομάτων std.
int main ()
{
int a, b = 3,
a = b;
α + = 2. // ισοδύναμο προς a = a + 2
cout << a;
}}
Επεξεργασία και εκτέλεση
Αύξηση και μείωση (++, -)
Ορισμένη έκφραση μπορεί να μειωθεί ακόμη περισσότερο: ο χειριστής αύξησης (++) και ο χειριστής μείωσης (-) αυξάνουν ή μειώνουν κατά μία την τιμή που είναι αποθηκευμένη σε μια μεταβλητή. Είναι ισοδύναμα με + = 1 και προς - = 1, αντίστοιχα. Ετσι:
++ x;
x + = 1.
x = χ + 1.
όλα ισοδυναμούν με τη λειτουργικότητά του. οι τρεις από αυτές αυξάνουν κατά μία τιμή του x. Στους πρώτους μεταγλωττιστές C, οι τρεις προηγούμενες εκφράσεις μπορεί να έχουν παραγάγει διαφορετικό εκτελέσιμο κώδικα ανάλογα με το ποιο από αυτά χρησιμοποιήθηκε. Σήμερα, αυτός ο τύπος βελτιστοποίησης κώδικα εκτελείται γενικά αυτόματα από τον μεταγλωττιστή, οπότε οι τρεις εκφράσεις πρέπει να παράγουν ακριβώς τον ίδιο εκτελέσιμο κώδικα.
Μια ιδιαιτερότητα αυτού του χειριστή είναι ότι μπορεί να χρησιμοποιηθεί τόσο ως πρόθεμα όσο και ως επίθημα. Αυτό σημαίνει ότι μπορεί να γραφτεί είτε πριν από το όνομα της μεταβλητής (++ x) είτε μετά από αυτό (x ++). Αν και σε απλές εκφράσεις όπως x ++ ή ++ x, και οι δύο έχουν ακριβώς το ίδιο νόημα. σε άλλες εκφράσεις στις οποίες αξιολογείται το αποτέλεσμα της διαδικασίας αύξησης ή μείωσης, μπορεί να έχουν σημαντική διαφορά ως προς το νόημά τους: Στην περίπτωση που ο χειριστής αύξησης χρησιμοποιείται ως πρόθεμα (++ x) της τιμής, η έκφραση αξιολογεί μέχρι την τελική τιμή του x, μόλις αυτή έχει ήδη αυξηθεί. Από την άλλη πλευρά, σε περίπτωση που χρησιμοποιείται ως επίθημα (x ++), η τιμή είναι επίσης αυξημένη, αλλά η έκφραση αξιολογείται στην τιμή που είχε το x πριν αυξηθεί. Παρατηρήστε τη διαφορά:
Παράδειγμα 1 Παράδειγμα 2
x = 3;
y = ++ χ.
// x περιέχει 4, y περιέχει
Οι χειριστές σχέσης και σύγκρισης (==,! =,>, <,> =, <=)
Δύο εκφράσεις μπορούν να συγκριθούν χρησιμοποιώντας χειριστές σχέσης και ισότητας. Για παράδειγμα, για να μάθουμε αν δύο τιμές είναι ίσες ή αν το ένα είναι μεγαλύτερο από το άλλο. Το αποτέλεσμα μιας τέτοιας ενέργειας είναι είτε αληθές είτε ψευδές (δηλ. Μια τιμή Boolean).
Οι σχεσιακοί χειριστές στη C ++ είναι:
περιγραφή του χειριστή
== Ίσο με
! = Δεν είναι ίση με
<Λιγότερο από
> Μεγαλύτερη από
<= Λιγότερο ή ίσο με
> = Μεγαλύτερη ή ίση με
Εδώ υπάρχουν μερικά παραδείγματα:
(7 == 5) // εκτιμάται ως ψευδής
(5> 4) // εκτιμάται ως αληθής
(3! = 2) // εκτιμάται ως αληθής

ΚαραλήςΓιώργος

(6> = 6) // εκτιμάται ως αληθής
(5 <5) // εκτιμάται ως ψευδής
Φυσικά, δεν είναι μόνο αριθμητικές σταθερές που μπορούν να συγκριθούν, αλλά μόνο οποιαδήποτε αξία, συμπεριλαμβανομένων, φυσικά, μεταβλητών. Ας υποθέσουμε ότι a = 2, b = 3 και c = 6, τότε:
(a == 5) // εκτιμάται ως ψευδής, αφού το a δεν είναι ίσο με 5
(a * b> = c) // αξιολογείται ως αληθής, αφού (2 * 3> = 6) είναι αληθές
(b + 4> a * c) // εκτιμάται ως ψευδής, αφού (3 + 4> 2 * 6) είναι ψευδής
((b = 2) == α) // αξιολογείται ως αληθής
Πρόσεχε! Ο χειριστής εκχώρησης (χειριστής =, με ένα ίση σημάδι) δεν είναι ο ίδιος με τον χειριστή σύγκρισης ισότητας (χειριστής ==, με δύο ίσες πινακίδες). η πρώτη (=) εκχωρεί την τιμή στα δεξιά της μεταβλητής στα αριστερά της, ενώ η άλλη (==) συγκρίνει εάν οι τιμές και στις δύο πλευρές του χειριστή είναι ίσες. Επομένως, στην τελευταία έκφραση ((b = 2) == a), εκχωρήσαμε αρχικά την τιμή 2 στο b και κατόπιν τη συγκρίνουμε με a (που αποθηκεύει επίσης την τιμή 2), αποδίδοντας true.
Λογικοί χειριστές (!, &&, ||)
Ο χειριστής ! είναι ο τελεστής C ++ για τη λειτουργία Boolean NOT. Έχει μόνο ένα τελεστή, στα δεξιά του, και το αντιστρέφει, παράγει ψευδή αν το όνομά του είναι αληθινό, και αληθινό αν ο τελεστής είναι ψευδής. Βασικά, επιστρέφει την αντίθετη Boolean τιμή αξιολόγησης του operand. Για παράδειγμα:
! (5 == 5) // εκτιμάται ως ψευδής επειδή η έκφραση στα δεξιά της (5 == 5) είναι αληθής
! (6 <= 4) // εκτιμάται ως αληθής επειδή (6 <= 4) θα ήταν ψευδές
! true // εκτιμάται ως ψευδής
// false // εκτιμάται ως αληθές
Οι λογικοί χειριστές && και || χρησιμοποιούνται όταν αξιολογούνται δύο εκφράσεις για να επιτευχθεί ένα μοναδικό σχεσιακό αποτέλεσμα. Ο χειριστής && αντιστοιχεί στη λογική λειτουργία Boolean AND, η οποία είναι αληθής αν και οι δύο τελεστές είναι αληθινοί και ψευδώς διαφορετικά. Ο ακόλουθος πίνακας δείχνει το αποτέλεσμα του χειριστή && και η αξιολόγηση της έκφρασης a && b:
&& ΦΟΡΕΑΣ (και)
a b a && b
αλήθεια αλήθεια αλήθεια
αληθινό ψεύτικο ψεύτικο
ψευδή αληθινό ψεύτικο
ψευδές ψεύτικο ψεύτικο
Ο χειριστής || αντιστοιχεί στη λογική λειτουργία Boolean Ή, η οποία αποδίδεται αληθής εάν ένας από τους τελεστές του είναι αληθής, οπότε είναι ψευδής μόνο όταν και οι δύο τελεστές είναι ψευδείς. Εδώ είναι τα πιθανά αποτελέσματα ενός || b:
|| ΕΠΙΧΕΙΡΗΣΗ (ή)
a b a || σι
αλήθεια αλήθεια αλήθεια
αληθινό ψεύτικο
ψευδή αλήθεια
ψευδές ψεύτικο ψεύτικο
Για παράδειγμα:
((5 == 5) && (3> 6)) // εκτιμάται σε false (true && false)
((5 == 5) || (3> 6)) // αξιολογείται ως true (true || false)
Όταν χρησιμοποιείτε τους λογικούς χειριστές, η C ++ αξιολογεί μόνο ό, τι είναι απαραίτητο από αριστερά προς τα δεξιά για να βγάλει το συνδυασμένο σχεσιακό αποτέλεσμα, αγνοώντας το υπόλοιπο. Επομένως, στο τελευταίο παράδειγμα ((5 == 5) || (3> 6)), η C ++ αξιολογεί πρώτα εάν

το 5 == 5 είναι αληθές και εάν ναι, ποτέ δεν ελέγχει εάν το 3> 6 είναι αληθές ή όχι. Αυτή η διαδικασία είναι γνωστή ως βραχυκύκλωμα και λειτουργεί έτσι για αυτούς τους χειριστές:
βραχυκύκλωμα χειριστή
&& αν η αριστερή πλευρά της έκφρασης είναι ψευδής, το συνδυασμένο αποτέλεσμα είναι ψευδές (η έκφραση της δεξιάς πλευράς δεν αξιολογείται ποτέ).
|| εάν η αριστερής πλευρά έκφραση είναι αληθής, το συνδυασμένο αποτέλεσμα είναι αληθές (η έκφραση της δεξιάς πλευράς δεν αξιολογείται ποτέ).
Αυτό είναι ιδιαίτερα σημαντικό όταν η δεξιά έκφραση έχει παρενέργειες, όπως οι μεταβαλλόμενες τιμές:
εάν ((i <10) && (++ i <n)) {/****} // σημειώστε ότι η συνθήκη αυξάνεται i
Εδώ, η συνδυασμένη έκφραση υπό όρους θα αυξανόταν από ένα, αλλά μόνο αν η κατάσταση στα αριστερά του && είναι αληθής, γιατί διαφορετικά, η συνθήκη στη δεξιά πλευρά (++ i <n) δεν αξιολογείται ποτέ.

Υποχρεωτικός τριμερής χειριστής (?)
Ο υποκείμενος χειριστής αξιολογεί μια έκφραση, επιστρέφοντας μία τιμή εάν αυτή η έκφραση αξιολογείται ως αληθής, και μια άλλη εάν η έκφραση αξιολογείται ως ψευδής. Η σύνταξή του είναι:
κατάσταση ? αποτέλεσμα1: αποτέλεσμα2
Εάν η συνθήκη είναι αληθής, ολόκληρη η έκφραση αξιολογείται ως αποτέλεσμα1, και διαφορετικά θα προκύψει2.
Δυναμικοί χειριστές (&, |, ^, ~, <<, >>)
Οι δυαδικοί χειριστές τροποποιούν τις μεταβλητές λαμβάνοντας υπόψη τα μοτίβα δυαδικών ψηφίων που αντιπροσωπεύουν τις τιμές που αποθηκεύουν.
περιγραφή ισοδύναμου χειριστή
& KAI Bitwise KAI
| Ή ΣΗΜΕΙΩΣΗ ΑΛΛΩΝ
^ XOR Σφάλμα αποκλειστικά OR
~ NOT Unary συμπλήρωμα (αναστροφή bit)
<< SHL Shift bits αριστερά
>> SHR Μετακινήστε τα κομμάτια δεξιά

36).Λειτουργία ρητής χύτευσης τύπου.

Οι χειριστές τύπου casting επιτρέπουν τη μετατροπή μιας τιμής ενός δεδομένου τύπου σε άλλο τύπο. Υπάρχουν διάφοροι τρόποι για να το κάνετε αυτό στην C ++. Το πιο απλό, το οποίο έχει κληρονομηθεί από τη γλώσσα C, είναι να προηγείται της έκφρασης προς μετατροπή από τον νέο τύπο που περικλείεται μεταξύ των παρενθέσεων (()):
int i;
float f = 3,14.
i = (int) f.
Ο προηγούμενος κώδικας μετατρέπει τον αριθμό κυμαινόμενου σημείου 3.14 σε ακέραια τιμή (3). το υπόλοιπο χάθηκε. Εδώ, ο χειριστής τύπου castcasting ήταν (int). Ένας άλλος τρόπος να κάνουμε το ίδιο πράγμα με τη C ++ είναι να χρησιμοποιήσουμε τη λειτουργική σημειογραφία που προηγείται της έκφρασης που πρέπει να μετατραπεί από τον τύπο και να περικλείει την έκφραση μεταξύ των παρενθέσεων:
i = int (f).
Και οι δύο τρόποι τύπων χύτευσης ισχύουν στην C ++.
μέγεθος του
Αυτός ο χειριστής δέχεται μία παράμετρο, η οποία μπορεί να είναι είτε ένας τύπος είτε μια μεταβλητή και επιστρέφει το μέγεθος σε bytes αυτού του τύπου ή αντικειμένου:

Επεξήγηση γλώσσας C++

x = μέγεθοςof (char);
Εδώ, η x έχει εκχωρηθεί η τιμή 1, επειδή char είναι ένας τύπος με μέγεθος ενός byte.
Η τιμή που επιστρέφεται από sizeof είναι μια σταθερά χρόνου μεταγλώττισης, οπότε καθορίζεται πάντα πριν από την εκτέλεση του προγράμματος.
Άλλοι χειριστές
Αργότερα σε αυτά τα μαθήματα, θα δούμε μερικούς ακόμα χειριστές, όπως αυτούς που αναφέρονται στους δείκτες ή τις ιδιαιτερότητες του αντικειμενοστρεφούς προγραμματισμού.
Προτεραιότητα των φορέων εκμετάλλευσης
Μία έκφραση μπορεί να έχει πολλαπλούς χειριστές. Για παράδειγμα:
x = 5 + 7% 2.
Στην C ++, η παραπάνω έκφραση εκχωρεί πάντοτε το 6 στη μεταβλητή x, επειδή ο χειριστής% έχει υψηλότερη προτεραιότητα από τον + χειριστή και πάντα αξιολογείται πριν. Τμήματα των εκφράσεων μπορούν να περικλείονται σε παρένθεση για να παρακάμψουν αυτήν την σειρά προτεραιότητας ή για να καταστήσουν σαφή την επιδιωκόμενη επίδραση. Παρατηρήστε τη διαφορά:
χ = 5 + (7% 2). // x = 6 (ίδιο και χωρίς παρένθεση)
x = (5 + 7)% 2, // x = 0
Από τη μέγιστη έως τη μικρότερη προτεραιότητα, οι χειριστές C ++ αξιολογούνται με την ακόλουθη σειρά:
Ομάδα προτεραιότητας Επίπεδο Περιγραφή χειριστή Ομαδοποίηση
1 Πεδίο εφαρμογής :: προσδιοριστής περιοχής από αριστερά προς δεξιά
2 Postfix (unary) ++ - αύξηση / μείωση postfix Από αριστερά προς δεξιά
() λειτουργικές μορφές
[] δείκτης
. -> πρόσβαση μελών
3 Πρόθεμα (unary) ++ - αύξηση πρόβλεψης / μείωση από δεξιά προς τα αριστερά
~ ~! δυαδική NOT / λογική NOT
+ - ενιαίο πρόθεμα
& * αναφορά / υποχώρηση
νέα διαγραφή κατανομής / ανακατανομής
sizeof πακέτου παραμέτρων
(τύπου) χύτευση τύπου στυλ C
4 Δείκτης-προς-μέλος. * -> * δείκτης πρόσβασης Από αριστερά προς τα δεξιά
5 Αριθμητική: κλιμάκωση * /% πολλαπλασιασμός, διαίρεση, μοντελο Από αριστερά προς δεξιά
6 Αριθμητική: προσθήκη + - προσθήκη, αφαίρεση Από αριστερά προς δεξιά
7 Μετατόπιση με στροφές << >> μετακίνηση αριστερά, μετακίνηση δεξιά Αριστερά προς τα δεξιά
8 Relational <> <=> = χειριστές σύγκρισης από αριστερά προς δεξιά
9 Ισότητα ==! = Ισότητα / ανισότητα Από αριστερά προς τα δεξιά
10 & & bitwise Αριστερά προς τα δεξιά
11 Αποκλειστική ή ^ δυαδική XOR Από αριστερά προς τα δεξιά
12 Συμπεριλαμβανομένων ή bitwise OR από αριστερά προς τα δεξιά
13 Σύζευξη && λογικό AND Από αριστερά προς τα δεξιά
14 Διαζύγιο || λογικό από αριστερά προς δεξιά
15 Εκφάνσεις επιπέδου εκχώρησης = * = / =% = + = - =
>> = << = & = ^ = | = αντιστοιχία εκχώρησης / σύνθεσης από δεξιά προς αριστερά
?: υπό όρους χειριστής
16 Sequencing, διαχωριστικό με κόμμα Από αριστερά προς τα δεξιά
Όταν μια έκφραση έχει δύο χειριστές με το ίδιο επίπεδο προτεραιότητας, η ομαδοποίηση καθορίζει ποιος αξιολογείται πρώτα: είτε από αριστερά προς δεξιά είτε από δεξιά προς τα αριστερά.

ΚαραλήςΓιώργος

Επεξήγηση γλώσσας C++

Η συμπλήρωση όλων των δευτερευόντων δηλώσεων σε παρένθεση (ακόμη και αυτές που δεν είναι απαραίτητες λόγω της προτεραιότητάς τους) βελτιώνει την αναγνωσιμότητα του κώδικα.
Βασική είσοδος / έξοδος(http://www.cplusplus.com/doc/tutorial/basic_io/)
Τα παραδείγματα προγραμμάτων των προηγούμενων ενοτήτων δεν παρέχουν επαρκή αλληλεπίδραση με τον χρήστη, αν υπάρχουν καθόλου. Απλώς τυπώνουν απλές τιμές στην οθόνη, αλλά η τυπική βιβλιοθήκη παρέχει πολλούς πρόσθετους τρόπους αλληλεπίδρασης με τον χρήστη μέσω των χαρακτηριστικών εισόδου / εξόδου του. Αυτή η ενότητα θα παρουσιάσει μια σύντομη εισαγωγή σε μερικές από τις πιο χρήσιμες.
Η C ++ χρησιμοποιεί μια βολική αφαίρεση που ονομάζεται ροή για την εκτέλεση εργασιών εισόδου και εξόδου σε διαδοχικά μέσα όπως η οθόνη, το πληκτρολόγιο ή ένα αρχείο. Μια ροή είναι μια οντότητα στην οποία ένα πρόγραμμα μπορεί να εισάγει ή να εξαγάγει χαρακτήρες από / προς. Δεν χρειάζεται να γνωρίζετε λεπτομέρειες σχετικά με τα μέσα που σχετίζονται με τη ροή ή με οποιαδήποτε από τις εσωτερικές της προδιαγραφές. Το μόνο που πρέπει να γνωρίζουμε είναι ότι τα ρεύματα είναι πηγή / προορισμός χαρακτήρων και ότι αυτοί οι χαρακτήρες παρέχονται / γίνονται δεκτοί διαδοχικά (δηλαδή, ο ένας μετά τον άλλο).
Η τυπική βιβλιοθήκη καθορίζει μια χούφτα αντικειμένων ροής που μπορούν να χρησιμοποιηθούν για την πρόσβαση σε ό, τι θεωρούνται οι τυπικές πηγές και οι προορισμοί χαρακτήρων από το περιβάλλον στο οποίο εκτελείται το πρόγραμμα:
περιγραφή ροής
cin τυπική ροή εισόδου
cout πρότυπη ροή εξόδου
cerr πρότυπο σφάλμα (έξοδος) ρεύμα
εμπλουτίζουν το τυπικό ρεύμα καταγραφής (εξόδου)
Θα δούμε με περισσότερες λεπτομέρειες μόνο το cout και το cin (η τυπική έξοδος και οι εισροές). ο cerr και το clog είναι επίσης ρεύματα εξόδου, έτσι ουσιαστικά λειτουργούν σαν cout, με τη μόνη διαφορά ότι αναγνωρίζουν ροές για συγκεκριμένους σκοπούς: μηνύματα σφάλματος και καταγραφή. που σε πολλές περιπτώσεις, στις περισσότερες ρυθμίσεις περιβάλλοντος, κάνουν ακριβώς το ίδιο πράγμα: εκτυπώνουν στην οθόνη, αν και μπορούν επίσης να αναπροσανατολιστούν ξεχωριστά.
Τυπική έξοδος (cout)
Στα περισσότερα περιβάλλοντα προγραμμάτων, η τυπική έξοδος από προεπιλογή είναι η οθόνη και το αντικείμενο ροής C ++ που έχει οριστεί για την πρόσβαση είναι cout.
Για μορφοποιημένες λειτουργίες εξόδου, το cout χρησιμοποιείται μαζί με τον χειριστή εισαγωγής, ο οποίος είναι γραμμένος ως << (δηλ. Δύο σήματα "λιγότερο από").

```
cout << "Παραπομπή παραγωγής". // εκτυπώνει την πρόταση εξόδου στην οθόνη
cout << 120; // εκτυπώνει τον αριθμό 120 στην οθόνη
cout << x; // εκτυπώνει την τιμή του x στην οθόνη
```

Ο χειριστής << εισάγει τα δεδομένα που τον ακολουθούν στην προηγούμενη ροή. Στα παραπάνω παραδείγματα, εισήγαγε τη φράση της εξόδου, τον αριθμό 120, και την τιμή της μεταβλητής x στην τυπική έξοδο ρεύματος εξόδου. Παρατηρήστε ότι η φράση στην πρώτη πρόταση περικλείεται σε διπλά εισαγωγικά (") επειδή είναι μια συμβολοσειρά, ενώ στο τελευταίο, το x δεν είναι Η διπλή παραπομπή είναι αυτό που κάνει τη διαφορά, όταν το κείμενο περικλείεται μεταξύ τους, το κείμενο εκτυπώνεται κυριολεκτικά • όταν δεν είναι, το κείμενο ερμηνεύεται ως το αναγνωριστικό μιας μεταβλητής και η αξία του εκτυπώνεται αντ 'αυτού Για παράδειγμα, αυτές οι δύο προτάσεις έχουν πολύ διαφορετικά αποτελέσματα:

```
cout << "Γειά σου"; // εκτυπώνει Γεια σας
cout << Γεια σας; // εκτυπώνει το περιεχόμενο της μεταβλητής Hello
```

Οι πολλαπλές λειτουργίες εισαγωγής (<<) μπορούν να αλυσοποιηθούν σε μία μόνο δήλωση:

```
cout << "Αυτή η" << "είναι μια" << "ενιαία δήλωση C ++".
```

ΚαραλήςΓιώργος

Επεξήγηση γλώσσας C++

Αυτή η τελευταία δήλωση θα εκτυπώσει το κείμενο Αυτή είναι μια ενιαία δήλωση C ++. Οι εισαγωγές αλυσίδας είναι ιδιαίτερα χρήσιμες για την ανάμειξη κυριολεκτικών και μεταβλητών σε μία και μόνη δήλωση:

cout << "Είμαι" << ηλικία << "ετών και ο ταχυδρομικός κώδικας μου είναι" << zipcode;

Αν υποθέσουμε ότι η μεταβλητή ηλικίας περιέχει την τιμή 24 και η μεταβλητή zipcode περιέχει 90064, η έξοδος της προηγούμενης εντολής θα είναι:

Είμαι 24 χρονών και ο ταχυδρομικός κώδικας μου είναι 90064

Αυτό που δεν γίνεται αυτόματα είναι να προσθέσετε διαλείμματα γραμμής στο τέλος, εκτός και αν σας ζητηθεί να το κάνετε. Για παράδειγμα, λάβετε τις ακόλουθες δύο δηλώσεις που εισάγονται σε cout:

cout << "Αυτή είναι μια πρόταση.";

cout << "Αυτή είναι μια άλλη πρόταση.";

Η έξοδος θα είναι σε μία γραμμή, χωρίς να υπάρχουν διαχωριστικά γραμμών μεταξύ τους. Κάτι όπως:

Αυτή είναι μια πρόταση. Αυτή είναι μια άλλη πρόταση.

Για να εισαχθεί ένα σπάσιμο γραμμής, πρέπει να εισαχθεί ένας χαρακτήρας νέας γραμμής στην ακριβή θέση που θα πρέπει να σπάσει η γραμμή. Στην C ++, ένας χαρακτήρας νέας γραμμής μπορεί να οριστεί ως \ n (δηλαδή ένας χαρακτήρας αντίστροφης κάθετος ακολουθούμενος από ένα πεζά n.). Για παράδειγμα:

cout << "Πρώτη πρόταση. \ n";

cout << "Δεύτερη πρόταση. \ nThird sentence.";

Αυτό παράγει την ακόλουθη έξοδο:

Πρώτη πρόταση.

Δεύτερη πρόταση.

Τρίτη πρόταση.

Εναλλακτικά, ο τελικός χειριστής μπορεί επίσης να χρησιμοποιηθεί για να σπάσει γραμμές. Για παράδειγμα:

cout << "Πρώτη πρόταση." << endl;

cout << "Δεύτερη πρόταση." << endl;

Αυτό θα έγραφε:

Πρώτη πρόταση.

Δεύτερη πρόταση.

Ο χειριστής endl παράγει ένα χαρακτήρα γραμμής νέας γραμμής, ακριβώς όπως κάνει η εισαγωγή του \ n ' αλλά έχει επίσης μια πρόσθετη συμπεριφορά: το buffer της ροής (εάν υπάρχει) ξεπλένεται, πράγμα που σημαίνει ότι η έξοδος απαιτείται να είναι γραμμένη φυσικά στη συσκευή, αν δεν ήταν ήδη. Αυτό επηρεάζει κυρίως τις πλήρως ρυθμισμένες ροές και το cout (γενικά) δεν είναι ένα πλήρες ρυθμισμένο ρεύμα. Ακόμα, είναι γενικά καλή ιδέα να χρησιμοποιείτε το endl μόνο όταν το ξεπλύνισμα του ρεύματος θα είναι ένα χαρακτηριστικό και '\ n' όταν δεν θα το έκανε. Είναι

cin >> α >> b;

Αυτό ισοδυναμεί με:

cin >> a;

cin >> b;

Και στις δύο περιπτώσεις, ο χρήστης αναμένεται να εισαγάγει δύο τιμές, μία για μεταβλητή a και άλλη για μεταβλητή b. Οποιοσδήποτε χώρος χρησιμοποιείται για τον διαχωρισμό δύο διαδοχικών λειτουργιών εισαγωγής. αυτό μπορεί να είναι είτε ένας χώρος, μια καρτέλα ή ένας χαρακτήρας νέας γραμμής.

cin και χορδές

Ο χειριστής εκχύλισης μπορεί να χρησιμοποιηθεί στον cin για να πάρει σειρές χαρακτήρων με τον ίδιο τρόπο όπως με τους βασικούς τύπους δεδομένων:

string mystring;

ΚαραλήςΓιώργος

Επεξήγηση γλώσσας C++

cin >> mystring;
Ωστόσο, η εξόρυξη cin θεωρεί πάντα ότι τα διαστήματα (whitespaces, tabs, new-line ...) τερματίζουν την αξία που εξάγεται και έτσι εξάγοντας ένα string να εξαγάγετε πάντα μια μόνο λέξη και όχι μια φράση ή μια ολόκληρη πρόταση.
Για να πάρουμε μια ολόκληρη γραμμή από τον cin, υπάρχει μια συνάρτηση, που ονομάζεται getline, η οποία παίρνει το ρεύμα (cin) ως πρώτο όρισμα, και η μεταβλητή συμβολοσειράς ως δεύτερη. Για παράδειγμα:
// cin με χορδές
#include <iostream>
#include <string>
χρησιμοποιώντας τον χώρο ονομάτων std.
int main ()
{
string mystr;
cout << "Ποιο είναι το όνομά σας;";
getline (cin, mystr);
cout << «Γεια σας» << mystr << ". \ n";
cout << "Ποια είναι η αγαπημένη σας ομάδα;";
getline (cin, mystr);
cout << "Μου αρέσει" << mystr << "επίσης! \ n";
επιστροφή 0?
}}
Ποιο είναι το όνομά σου? Ομηρος Σίμψον
Γεια σας Όμηρος Σίμψονς.
Ποια είναι η αγαπημένη σου ομάδα? Ολυμπιακός
Μόνο Ολυμπιακός!
Επεξεργασία και εκτέλεση
Παρατηρήστε πώς και στις δύο κλήσεις σε getline, χρησιμοποιήσαμε το ίδιο αναγνωριστικό συμβολοσειράς (mystr). Αυτό που κάνει το πρόγραμμα στη δεύτερη κλήση απλώς αντικαθιστά το προηγούμενο περιεχόμενο με το νέο που εισάγεται.
Η τυπική συμπεριφορά που οι περισσότεροι χρήστες αναμένουν από ένα πρόγραμμα κονσόλας είναι ότι κάθε φορά που το πρόγραμμα ζητά από τον χρήστη την είσοδο, ο χρήστης εισάγει το πεδίο και, στη συνέχεια, πατά ENTER (ή RETURN). Δηλαδή, η είσοδος γενικά αναμένεται να συμβεί από την άποψη των γραμμών στα προγράμματα κονσόλας, και αυτό μπορεί να επιτευχθεί με τη χρήση του getline για την απόκτηση εισόδου από τον χρήστη. Επομένως, αν δεν έχετε έναν ισχυρό λόγο να μην κάνετε, θα πρέπει πάντα να χρησιμοποιείτε getline για να λάβετε εισροή στα προγράμματα κονσόλας αντί να εξαγάγετε από το cin.
stringstream
Η τυπική κεφαλίδα <sstream> ορίζει έναν τύπο που ονομάζεται stringstream που επιτρέπει σε μια συμβολοσειρά να αντιμετωπιστεί ως ροή, επιτρέποντας έτσι την εξαγωγή ή την εισαγωγή πράξεων από / σε συμβολοσειρές με τον ίδιο τρόπο που εκτελούνται στο cin και cout. Αυτή η λειτουργία είναι πολύ χρήσιμη για τη μετατροπή των συμβολοσειρών σε αριθμητικές τιμές και αντίστροφα. Για παράδειγμα, για να εξαγάγουμε έναν ακέραιο από μια συμβολοσειρά μπορούμε να γράψουμε:
string mystr ("1204");
int myint;
stringstream (mystr) >> myint;
Αυτό δηλώνει μια συμβολοσειρά με αρχικοποιημένη τιμή "1204" και μια μεταβλητή τύπου int. Στη συνέχεια, η τρίτη γραμμή χρησιμοποιεί αυτή τη μεταβλητή για να εξαχθεί από ένα stringstream που κατασκευάζεται από τη συμβολοσειρά. Αυτό το κομμάτι του κώδικα αποθηκεύει την αριθμητική τιμή 1204 στη μεταβλητή που ονομάζεται myint.

ΚαραλήςΓιώργος

/ stringstreams
#include <iostream>
#include <string>
#include <sstream>
χρησιμοποιώντας τον χώρο ονομάτων std.
int main ()
{
string mystr;
τιμή float = 0.
int ποσότητα = 0;
cout << "Εισάγετε τιμή:";
getline (cin, mystr);
stringstream (mystr) >> τιμή;
cout << "Εισάγετε ποσότητα:";
getline (cin, mystr);
stringstream (mystr) >> ποσότητα;
cout << "Συνολική τιμή:" << τιμή * ποσότητα << endl;
επιστροφή 0?
}}
Καταχώρηση τιμής: 22.25
Εισάγετε ποσότητα: 7
Συνολική τιμή: 155.75
Επεξεργασία και εκτέλεση
Σε αυτό το παράδειγμα, αποκτάμε έμμεσες αριθμητικές τιμές από την τυπική είσοδο: Αντί να εξαγάγουμε αριθμητικές τιμές απευθείας από τον cin, παίρνουμε γραμμές από αυτό σε ένα αντικείμενο string (mystr), και στη συνέχεια εξαγάγουμε τις τιμές από αυτή την συμβολοσειρά στην τιμή μεταβλητών και ποσότητα. Μόλις αυτές είναι αριθμητικές τιμές, μπορούν να εκτελούνται αριθμητικές πράξεις πάνω τους, όπως το πολλαπλασιασμό τους για να αποκτήσετε μια συνολική τιμή.
Με αυτήν την προσέγγιση να πάρουμε ολόκληρες γραμμές και να εξαγάγουμε το περιεχόμενό τους, διαχωρίζουμε τη διαδικασία της εισαγωγής των χρηστών από την ερμηνεία τους ως δεδομένων, επιτρέποντας έτσι τη διαδικασία εισαγωγής να είναι αυτό που ο χρήστης αναμένει και ταυτόχρονα να αποκτήσει μεγαλύτερο έλεγχο στη μετατροπή του περιεχομένου σε χρήσιμα δεδομένα από το πρόγραμμα.
ΔΟΜΗ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ

37).Δηλώσεις και έλεγχος ροής.

Μια απλή εντολή C ++ είναι καθεμία από τις ξεχωριστές οδηγίες ενός προγράμματος, όπως οι μεταβλητές δηλώσεις και εκφράσεις που εμφανίζονται σε προηγούμενες ενότητες. Πάντα τελειώνουν με ένα ερωτηματικό (?), Και εκτελούνται με την ίδια σειρά με την οποία εμφανίζονται σε ένα πρόγραμμα.
Αλλά τα προγράμματα δεν περιορίζονται σε μια γραμμική ακολουθία δηλώσεων. Κατά τη διάρκεια της διαδικασίας, ένα πρόγραμμα μπορεί να επαναλάβει τμήματα του κώδικα ή να λάβει αποφάσεις και να διχαστεί. Για το σκοπό αυτό, η C ++ παρέχει δηλώσεις ελέγχου ροής που χρησιμεύουν για να καθορίσουν τι πρέπει να γίνει από το πρόγραμμά μας, πότε και υπό ποιες συνθήκες.
Πολλές από τις δηλώσεις ελέγχου ροής που εξηγούνται σε αυτήν την ενότητα απαιτούν μια γενική (υπο) δήλωση ως μέρος της σύνταξής της. Αυτή η δήλωση μπορεί να είναι είτε μια απλή δήλωση C ++, -όπως μια ενιαία εντολή, τερματισμένη με ένα ερωτηματικό (?) - ή μια σύνθετη δήλωση. Μια σύνθετη δήλωση είναι μια ομάδα δηλώσεων (καθεμία από τις οποίες τερματίζεται

από το δικό της ερωτηματικό), αλλά όλες ομαδοποιημένες σε ένα μπλοκ, που περικλείονται σε σγουρά: {}:

{statement1; δήλωση2; δήλωση3; }}

Το σύνολο του μπλοκ θεωρείται μια ενιαία δήλωση (που αποτελείται από πολλαπλές υποσημειώσεις). Κάθε φορά που μια γενική δήλωση είναι μέρος της σύνταξης μιας εντολής ελέγχου ροής, αυτό μπορεί να είναι είτε μια απλή δήλωση είτε μια σύνθετη δήλωση.

Δηλώσεις επιλογής: αν και αλλιώς

Η λέξη κλειδί if χρησιμοποιείται για την εκτέλεση μιας δήλωσης ή ενός μπλοκ, εάν και μόνο εάν πληρούται μία προϋπόθεση. Η σύνταξή του είναι:

εάν (δήλωση)

Εδώ, η συνθήκη είναι η έκφραση που αξιολογείται. Εάν αυτή η συνθήκη είναι αληθής, εκτελείται δήλωση. Αν είναι ψευδής, η δήλωση δεν εκτελείται (απλά αγνοείται) και το πρόγραμμα συνεχίζεται αμέσως μετά από ολόκληρη τη δήλωση επιλογής.

Για παράδειγμα, το παρακάτω κομμάτι κώδικα εκτυπώνει το μήνυμα (x είναι 100), μόνο εάν η τιμή που είναι αποθηκευμένη στη μεταβλητή x είναι πράγματι 100:

αν (x == 100)

cout << "x είναι 100".

Εάν το x δεν είναι ακριβώς 100, αυτή η δήλωση αγνοείται και τίποτα δεν εκτυπώνεται.

Αν θέλετε να συμπεριλάβετε περισσότερες από μία δήλωση που θα εκτελεστούν όταν πληρούται η προϋπόθεση, αυτές οι δηλώσεις πρέπει να περικλείονται σε τιράντες ({}), που σχηματίζουν ένα μπλοκ:

αν (x == 100)

{

cout << "x είναι";

cout << x;

}}

Ως συνήθως, οι παρεμβολές και οι διαχωρισμοί γραμμών στον κώδικα δεν έχουν καμία επίδραση, οπότε ο παραπάνω κωδικός είναι ισοδύναμος με:

αν (x == 100) {cout << "x είναι"; cout << x; }}

Οι δηλώσεις επιλογής με το if μπορούν επίσης να καθορίσουν τι συμβαίνει όταν η προϋπόθεση δεν πληρούται, χρησιμοποιώντας την άλλη λέξη-κλειδί για να εισαγάγετε μια εναλλακτική δήλωση. Η σύνταξή του είναι:

if (κατάσταση) statement1 else statement2

όπου η εντολή1 εκτελείται στην περίπτωση που η προϋπόθεση είναι αληθής, και σε περίπτωση που δεν είναι, εκτελείται η εντολή2.

Για παράδειγμα:

αν (x == 100)

cout << "x είναι 100".

αλλού

cout << "x δεν είναι 100".

Αυτό εκτυπώνει το x είναι 100, αν πράγματι το x έχει τιμή 100, αλλά αν δεν το κάνει και μόνο αν δεν το εκτυπώνει, το x δεν είναι 100 αντί.

Αρκετές αν δομές + άλλο μπορούν να συνενωθούν με σκοπό να ελέγξουν ένα εύρος τιμών. Για παράδειγμα:

αν (x> 0)

cout << "x είναι θετικό".

else αν (x <0)

cout << "x είναι αρνητικό";

αλλού

cout << "x είναι 0".

Αυτό εκτυπώνει εάν το x είναι θετικό, αρνητικό ή μηδενικό με συνένωση δύο δομών if-else. Και πάλι, θα ήταν δυνατό να εκτελεστούν περισσότερες από μία μόνο δήλωση ανά περίπτωση με την ομαδοποίησή τους σε μπλοκ που περικλείονται σε τιράντες: {}.

Δηλώσεις επανάληψης (βρόχοι)

Οι βρόχοι επαναλαμβάνουν μια δήλωση αρκετές φορές ή όταν πληρούται μια προϋπόθεση. Εισάγονται από τις λέξεις-κλειδιά ενώ κάνετε, και για.

Ο βρόχος ενώ

Το απλούστερο είδος βρόχου είναι ο βρόχος while-loop. Η σύνταξή του είναι:

ενώ η δήλωση (έκφραση)

Το while-loop απλά επαναλαμβάνει τη δήλωση ενώ η έκφραση είναι αληθής. Εάν, μετά από οποιαδήποτε εκτέλεση μιας δήλωσης, η έκφραση δεν είναι πλέον αληθής, ο βρόχος τελειώνει και το πρόγραμμα συνεχίζεται αμέσως μετά τον βρόχο. Για παράδειγμα, ας ρίξουμε μια ματιά σε μια αντίστροφη μέτρηση χρησιμοποιώντας ένα while-loop:

// custom countdown using while
#include <iostream>
χρησιμοποιώντας τον χώρο ονομάτων std.
int main ()
{
int n = 10;
ενώ (n> 0) {
cout << n << ",";
- n;
}}
cout << "liftoff! \ n";
}}

10, 9, 8, 7, 6, 5, 4, 3, 2, 1, liftoff!

Επεξεργασία και εκτέλεση

Η πρώτη δήλωση στις κύριες ομάδες n σε τιμή 10. Αυτός είναι ο πρώτος αριθμός στην αντίστροφη μέτρηση. Στη συνέχεια ξεκινάει ο βρόχος while: αν αυτή η τιμή πληροί την συνθήκη n> 0 (το n είναι μεγαλύτερο από το μηδέν), τότε το μπλοκ που ακολουθεί την κατάσταση εκτελείται και επαναλαμβάνεται για όσο διάστημα η συνθήκη (n> 0) παραμένει αληθής.

Η όλη διαδικασία του προηγούμενου προγράμματος μπορεί να ερμηνευτεί σύμφωνα με το ακόλουθο σενάριο (ξεκινώντας από το κύριο):

n έχει εκχωρηθεί μια τιμή

Η συνθήκη while (n> 0). Σε αυτό το σημείο υπάρχουν δύο δυνατότητες:

η προϋπόθεση είναι αληθής: η δήλωση εκτελείται (στο βήμα 3)

η συνθήκη είναι ψευδής: αγνοήστε τη δήλωση και συνεχίστε μετά από αυτήν (στο βήμα 5)

Εκτέλεση δήλωσης:

cout << n << ",";
- n;

(εκτυπώνει την τιμή)

Συμπεριέχει ένα βρόχο, εκτός από το ότι η κατάσταση αξιολογείται μετά την εκτέλεση της δήλωσης αντί για πριν, εξασφαλίζοντας τουλάχιστον μία εκτέλεση δήλωσης, ακόμη και αν δεν πληρούται ποτέ η προϋπόθεση. Για παράδειγμα, το ακόλουθο παράδειγμα προγράμματος επαναλαμβάνει κάθε κείμενο που εισάγει ο χρήστης μέχρι να εισέλθει ο χρήστης αντίο:

// echo μηχάνημα
#include <iostream>
#include <string>
χρησιμοποιώντας τον χώρο ονομάτων std.
int main ()
{

str string;
κάνω {
cout << "Εισάγετε κείμενο:";
getline (cin, str);
cout << "Πληκτρολογήσατε:" << str << '\ n';
} ενώ (str! = "αντίο")?
}}
Εισαγάγετε κείμενο: γεια
Εισήγατε: γεια
Εισαγάγετε κείμενο: ποιος είναι εκεί;
Είσαστε: ποιος είναι εκεί;
Εισαγάγετε κείμενο: αντίο
Εισήγατε: αντίο
Επεξεργασία και εκτέλεση
Ο βρόχος do-while προτιμάται συνήθως κατά τη διάρκεια ενός βρόχου ενώ η δήλωση πρέπει να εκτελεστεί τουλάχιστον μία φορά, όπως όταν η κατάσταση που ελέγχεται στο τέλος του βρόχου καθορίζεται από την ίδια τη δήλωση βρόχου. Στο προηγούμενο παράδειγμα, η είσοδος χρήστη μέσα στο μπλοκ είναι αυτό που θα καθορίσει εάν τελειώνει ο βρόχος. Και έτσι, ακόμα και αν ο χρήστης θέλει να τερματίσει το βρόχο το συντομότερο δυνατόν εισάγοντας αντίο, το μπλοκ στον βρόχο πρέπει να εκτελεστεί τουλάχιστον μία φορά για να ζητήσει την είσοδο και η κατάσταση μπορεί στην πραγματικότητα να καθοριστεί μόνο μετά εκτελείται.
Ο βρόχος για
Ο βρόχος for έχει σχεδιαστεί για να επαναλαμβάνει πολλές φορές. Η σύνταξή του είναι:
για την εντολή (προετοιμασία, κατάσταση, αύξηση)
Όπως και ο βρόχος while, αυτό το βρόχο επαναλαμβάνει τη δήλωση ενώ η προϋπόθεση είναι αληθής. Αλλά, επιπλέον, ο βρόχος for παρέχει συγκεκριμένες θέσεις για να περιέχει μια έκφραση αρχικοποίησης και αύξησης, που εκτελείται πριν ξεκινήσει ο βρόχος για πρώτη φορά και μετά από κάθε επανάληψη, αντίστοιχα. Ως εκ τούτου, είναι ιδιαίτερα χρήσιμο να χρησιμοποιηθούν οι μεταβλητές ως προϋπόθεση.
Λειτουργεί με τον ακόλουθο τρόπο:
εκτελείται αρχικοποίηση. Γενικά, αυτό δηλώνει μια μεταβλητή μεταβλητή και την ορίζει σε κάποια αρχική τιμή. Αυτό εκτελείται μία φορά, στην αρχή του βρόχου.
κατάσταση. Αν είναι αλήθεια, ο βρόχος συνεχίζει. διαφορετικά, ο βρόχος τελειώνει και η πρόταση παραλείπεται, πηγαίνοντας απευθείας στο βήμα 5.
δήλωση εκτελείται. Ως συνήθως, μπορεί να είναι είτε μία δήλωση είτε ένα μπλοκ που περικλείεται σε σγουρά {}}.
αυξάνεται η εκτέλεση και ο βρόχος επανέρχεται στο βήμα 2.
ο βρόχος τελειώνει: η εκτέλεση συνεχίζεται από την επόμενη δήλωση μετά από αυτήν.
Εδώ είναι το παράδειγμα αντίστροφης μέτρησης χρησιμοποιώντας ένα βρόχο for:
// αντίστροφη μέτρηση χρησιμοποιώντας ένα για βρόχο
#include <iostream>
χρησιμοποιώντας τον χώρο ονομάτων std.
int main ()
{
για (int n = 10, n> 0, n--) {
cout << n << ",";
}}
cout << "liftoff! \ n";
}}
10, 9, 8, 7, 6, 5, 4, 3, 2, 1, liftoff!
Επεξεργασία και εκτέλεση

Επεξήγηση γλώσσας C++

Τα τρία πεδία σε ένα for-loop είναι προαιρετικά. Μπορούν να αφεθούν κενά, αλλά σε όλες τις περιπτώσεις απαιτούνται τα σημάδια τοξοειδούς μεταξύ τους. Για παράδειγμα, για το (; n <10;) είναι ένας βρόχος χωρίς αρχικοποίηση ή αύξηση (ισοδύναμος με ένα βρόχο ενώ). και για το (n <10; ++ n) είναι ένας βρόχος με αύξηση, αλλά καμία αρχικοποίηση (ίσως επειδή η μεταβλητή είχε ήδη αρχικοποιηθεί πριν από τον βρόχο). Ένας βρόχος χωρίς όρους είναι ισοδύναμος με έναν βρόχο με αληθινή συνθήκη (δηλ. Έναν άπειρο βρόχο).

Επειδή καθένα από τα πεδία εκτελείται σε μια συγκεκριμένη χρονική στιγμή στον κύκλο ζωής ενός βρόχου, μπορεί να είναι χρήσιμο να εκτελέσετε περισσότερες από μία μόνο έκφραση ως οποιαδήποτε εκκίνηση, προϋπόθεση ή δήλωση. Δυστυχώς, αυτά δεν είναι δηλώσεις, αλλά μάλλον απλές εκφράσεις και επομένως δεν μπορούν να αντικατασταθούν από ένα μπλοκ. Ως εκφράσεις, ωστόσο, μπορούν να χρησιμοποιήσουν τον χειριστή των παραγράφων (,): Αυτός ο χειριστής είναι ένας διαχωριστής έκφρασης και μπορεί να διαχωρίσει πολλαπλές εκφράσεις όπου γενικά αναμένεται μόνο μία. Για παράδειγμα, με τη χρήση του, θα ήταν δυνατό για ένα for loop να χειριστεί δύο μεταβλητές μετρητή, αρχικοποιώντας και αυξάνοντας και τα δύο:

(η = 0, ί = 100, η! = i, ++ η, - i)
{
// οπουδήποτε εδώ ...
}}

Αυτός ο βρόχος θα εκτελεστεί 50 φορές εάν ούτε το n ούτε το i δεν έχουν τροποποιηθεί εντός του βρόχου:

n ξεκινά με μια τιμή 0 και i με 100, η συνθήκη είναι n! = i (δηλ., ότι το n δεν είναι ίσο με το i). Επειδή το n αυξάνεται κατά ένα και μειώνεται κατά ένα σε κάθε επανάληψη, η κατάσταση του βρόχου θα γίνει ψευδής μετά την 50ή επανάληψη, όταν τόσο το n όσο και το i είναι ίσο με 50.

Περιοχή βάσει βρόχου

Ο for-loop έχει μια άλλη σύνταξη, η οποία χρησιμοποιείται αποκλειστικά με σειρές:

για τη δήλωση (εύρος).

Αυτό το είδος βρόχου επαναλαμβάνεται πάνω από όλα τα στοιχεία της εμβέλειας, όπου η δήλωση δηλώνει κάποια μεταβλητή ικανή να πάρει την αξία ενός στοιχείου σε αυτό το εύρος. Οι κλίμακες είναι ακολουθίες στοιχείων, συμπεριλαμβανομένων συστοιχιών, δοχείων και οποιουδήποτε άλλου τύπου που υποστηρίζει τις λειτουργίες αρχίζουν και τελειώνουν. Οι περισσότεροι από αυτούς τους τύπους δεν έχουν εισαχθεί ακόμη σε αυτό το σεμινάριο, αλλά έχουμε ήδη εξοικειωθεί με τουλάχιστον ένα είδος σειράς: χορδές, οι οποίες είναι ακολουθίες χαρακτήρων.

Ένα παράδειγμα της σειράς που βασίζεται για το βρόχο χρησιμοποιώντας τις συμβολοσειρές:

// με βάση το εύρος για βρόχο
#include <iostream>
#include <string>
χρησιμοποιώντας τον χώρο ονομάτων std.
int main ()
{
The goto statement

το goto επιτρέπει να κάνετε ένα απόλυτο άλμα σε ένα άλλο σημείο του προγράμματος. Αυτό το απεριόριστο άλμα αγνοεί τα επίπεδα φωλιάσματος και δεν προκαλεί την αυτόματη ξεδίπλωση της στοίβας. Επομένως, είναι ένα χαρακτηριστικό γνώρισμα που πρέπει να χρησιμοποιείται με προσοχή και κατά προτίμηση μέσα στο ίδιο μπλοκ δηλώσεων, ειδικά παρουσία τοπικών μεταβλητών.The destination point is identified by a label, which is then used as an argument for the goto statement. A label is made of a valid identifier followed by a colon (:).

το goto θεωρείται γενικά χαρακτηριστικό χαμηλού επιπέδου, χωρίς ιδιαίτερες περιπτώσεις χρήσης σε σύγχρονα παραδείγματα προγραμματισμού υψηλότερου επιπέδου που χρησιμοποιούνται γενικά με τη C ++. Αλλά, σαν παράδειγμα, εδώ είναι μια έκδοση του βρόχου

ΚαραλήςΓιώργος

αντίστροφης μέτρησης χρησιμοποιώντας το goto:
// παράδειγμα για το βρόχο
#include <iostream>
χρησιμοποιώντας τον χώρο ονομάτων std.

int main ()
{
int n = 10;
mylabel:
cout << n << ",";
n-;
αν (n> 0) goto mylabel.
cout << liftoff! \ n ";
} 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, liftoff!
Άλλη πρόταση επιλογής: διακόπτης.
Η σύνταξη της δήλωσης διακόπτη είναι λίγο περίεργη. Σκοπός του είναι να ελέγξει για μια τιμή μεταξύ μιας σειράς πιθανών σταθερών εκφράσεων. Είναι κάτι παρόμοιο με τη σύζευξη των δηλώσεων if-else, αλλά περιορίζεται στις συνεχείς εκφράσεις. Η πιο συνηθισμένη σύνταξη είναι:
διακόπτης (έκφραση)
{
περίπτωση σταθερά1:
ομάδα-των-δηλώσεων-1?
Διακοπή;
περίπτωση σταθεράς2:
ομάδα-των-δηλώσεων-2?
Διακοπή;
.
.
.
Προκαθορισμένο:
default-group of-statements
}}

Λειτουργεί με τον ακόλουθο τρόπο: ο διακόπτης αξιολογεί την έκφραση και ελέγχει αν είναι ισοδύναμος με την constant1. αν είναι, εκτελεί την ομάδα-των-statements-1 μέχρι να βρει τη δήλωση break. Όταν εντοπίσει αυτή τη δήλωση διακοπής, το πρόγραμμα μεταβαίνει στο τέλος ολόκληρης της εντολής διακόπτη (το στήριγμα κλεισίματος).
Εάν η έκφραση δεν ήταν ίση με την σταθερά1, τότε ελέγχεται έναντι της σταθεράς2. Αν είναι ίσο με αυτό, εκτελεί την ομάδα-statements-2 μέχρι να βρεθεί ένα διάλειμμα, όταν μεταπηδά στο τέλος του διακόπτη.

Τέλος, αν η τιμή της έκφρασης δεν ταιριάζει με καμία από τις σταθερές που έχουν οριστεί προηγουμένως (μπορεί να υπάρχει οποιοσδήποτε αριθμός από αυτές), το πρόγραμμα εκτελεί τις δηλώσεις που περιλαμβάνονται μετά την προεπιλεγμένη: ετικέτα, αν υπάρχει (εφόσον είναι προαιρετική).

Και τα δύο ακόλουθα θραύσματα κώδικα έχουν την ίδια συμπεριφορά, αποδεικνύοντας το if-else ισοδύναμο μιας εντολής διακόπτη:
παράδειγμα διακόπτη εάν -αν άλλο ισοδύναμο
διακόπτης (x) {
περίπτωση 1:

cout << "x είναι 1".
Διακοπή;
περίπτωση 2:
cout << "το x είναι 2".
Διακοπή;
Προκαθορισμένο:
cout << "τιμή του x άγνωστου";
} αν (x == 1) {
cout << "x είναι 1".
}}
else αν (x == 2) {
cout << "το x είναι 2".
}}
αλλού {
cout << "τιμή του x άγνωστου";
}}

Η εντολή διακόπτη έχει μια κάπως περίεργη σύνταξη που κληρονόμησε από τους πρώτους χρόνους των πρώτων μεταγλωττιστών Γ, επειδή χρησιμοποιεί ετικέτες αντί μπλοκ. Στην πιο συνηθισμένη χρήση (που φαίνεται παραπάνω), αυτό σημαίνει ότι απαιτούνται δηλώσεις σπασίματος μετά από κάθε ομάδα δηλώσεων για μια συγκεκριμένη ετικέτα. Εάν δεν συμπεριλαμβάνεται το σπάσιμο, εκτελούνται επίσης όλες οι δηλώσεις που ακολουθούν την περίπτωση (συμπεριλαμβανομένων εκείνων που βρίσκονται κάτω από οποιεσδήποτε άλλες ετικέτες) μέχρι να φτάσει το τέλος του μπλοκ διακόπτη ή μια εντολή άλματος (όπως break). Εάν το παραπάνω παράδειγμα δεν είχε τη δήλωση σπάσει μετά την πρώτη ομάδα για την πρώτη περίπτωση, το πρόγραμμα δεν θα μεταπηδήσει αυτόματα στο τέλος του μπλοκ διακόπτη μετά την εκτύπωση x είναι 1 και αντίθετα θα συνέχιζε να εκτελεί τις δηλώσεις στην περίπτωση 2 είναι 2). Στη συνέχεια, θα συνεχίσει να το κάνει μέχρι να εμφανιστεί μια δήλωση διακοπής ή το τέλος του μπλοκ διακόπτη. Αυτό καθιστά περιττό να περιληφθούν οι δηλώσεις για κάθε περίπτωση σε braces {} και μπορεί επίσης να είναι χρήσιμο να εκτελεστεί η ίδια ομάδα δηλώσεων για διαφορετικές πιθανές τιμές. Για παράδειγμα:

διακόπτης (x) {
περίπτωση 1:
περίπτωση 2:
περίπτωση 3:
cout << "το χ είναι 1, 2 ή 3".
Διακοπή;
Προκαθορισμένο:
cout << "x δεν είναι 1, 2 ούτε 3".
}}

Παρατηρήστε ότι ο διακόπτης περιορίζεται για να συγκρίνετε την εκτιμώμενη έκφρασή του με ετικέτες που είναι σταθερές εκφράσεις. Δεν είναι δυνατή η χρήση μεταβλητών ως ετικέτες ή εύρους τιμών, επειδή δεν είναι έγκυρες συνεχείς εκφράσεις C ++.
Για να ελέγξετε για εύρη τιμών ή τιμές που δεν είναι σταθερές, είναι προτιμότερο να χρησιμοποιήσετε τις αλληλουχίες των if και else if statements.

38).Λειτουργίες

Οι λειτουργίες επιτρέπουν τη δομή προγραμμάτων σε τμήματα του κώδικα για την εκτέλεση μεμονωμένων εργασιών.

Επεξήγηση γλώσσας C++

Στην C ++, μια συνάρτηση είναι μια ομάδα δηλώσεων που δίνεται ένα όνομα και που μπορεί να καλείται από κάποιο σημείο του προγράμματος. Η πιο συνηθισμένη σύνταξη για τον ορισμό μιας συνάρτησης είναι:

όνομα τύπου (παράμετρος 1, παράμετρος2, ...) {δηλώσεις}

Που:

- ο τύπος είναι ο τύπος της τιμής που επιστρέφεται από τη συνάρτηση.
- όνομα είναι το αναγνωριστικό με το οποίο μπορεί να καλείται η λειτουργία.
- παραμέτρους (όσες χρειάζονται): Κάθε παράμετρος αποτελείται από έναν τύπο ακολουθούμενο από ένα αναγνωριστικό, με κάθε παράμετρο να διαχωρίζεται από το επόμενο με κόμμα. Κάθε παράμετρος μοιάζει πολύ με μια κανονική δήλωση μεταβλητής (για παράδειγμα: int x), και στην πραγματικότητα ενεργεί μέσα στη συνάρτηση ως κανονική μεταβλητή που είναι τοπική για τη λειτουργία. Ο σκοπός των παραμέτρων είναι να επιτρέπεται η διέλευση των επιχειρημάτων από τη θέση από την οποία καλείται.
- δηλώσεις είναι το σώμα της λειτουργίας. Είναι ένα μπλοκ δηλώσεων που περιβάλλεται από τιράντες {} που καθορίζουν τι πραγματικά κάνει η λειτουργία.

Ας ρίξουμε μια ματιά σε ένα παράδειγμα:

```
// παράδειγμα λειτουργίας
#include <iostream>
χρησιμοποιώντας τον χώρο ονομάτων std.
int προσθήκη (int a, int b)
{
int r;
r = a + b.
επιστροφή r;
}}

int main ()
{
int z;
z = προσθήκη (5,3).
cout << "Το αποτέλεσμα είναι" << z;
}}
```

Το αποτέλεσμα είναι 8

Επεξεργασία και εκτέλεση

Αυτό το πρόγραμμα χωρίζεται σε δύο λειτουργίες: προσθήκη και κύρια. Να θυμάστε ότι ανεξάρτητα από τη σειρά με την οποία ορίζονται, ένα πρόγραμμα C ++ αρχίζει πάντα καλώντας το main. Στην πραγματικότητα, η κύρια είναι η μόνη λειτουργία που ονομάζεται αυτόματα και ο κώδικας σε οποιαδήποτε άλλη λειτουργία εκτελείται μόνο αν η λειτουργία του καλείται από το κύριο (άμεσα ή έμμεσα).

Στο παραπάνω παράδειγμα, το κύριο ξεκινά με τη δήλωση της μεταβλητής z του τύπου int, και αμέσως μετά, εκτελεί την πρώτη κλήση λειτουργίας: καλεί την προσθήκη. Η κλήση σε μια λειτουργία ακολουθεί μια δομή παρόμοια με τη δήλωσή της. Στο παραπάνω παράδειγμα, η πρόσκληση για προσθήκη μπορεί να συγκριθεί με τον ορισμό της με λίγες γραμμές νωρίτερα:

Οι παράμετροι στη δήλωση λειτουργίας έχουν σαφή αντιστοιχία με τα επιχειρήματα που πέρασαν στην κλήση λειτουργίας. Η κλήση περνά δύο τιμές, 5 και 3, στη λειτουργία. αυτά αντιστοιχούν στις παραμέτρους a και b, που δηλώνονται για την προσθήκη λειτουργίας.

Στο σημείο στο οποίο καλείται η λειτουργία από το κύριο, ο έλεγχος μεταβιβάζεται σε προσθήκη λειτουργίας: εδώ, η εκτέλεση του κύριου διακόπτεται και θα επαναληφθεί μόνο όταν ολοκληρωθεί η λειτουργία προσθήκης. Τη στιγμή της κλήσης λειτουργίας, η τιμή και των δύο παραμέτρων (5 και 3) αντιγράφεται στις τοπικές μεταβλητές int a και int b μέσα στη συνάρτηση.

ΚαραλήςΓιώργος

Στη συνέχεια, μέσα στην προσθήκη, μια άλλη τοπική μεταβλητή δηλώνεται (int r), και με τη βοήθεια της έκφρασης r = a + b, το αποτέλεσμα ενός συν το b έχει εκχωρηθεί στο r; η οποία, για αυτή την περίπτωση, όπου a είναι 5 και b είναι 3, σημαίνει ότι το 8 αποδίδεται στο r.

39). Ενσωματωμένες Λειτουργίες

Η κλήση μιας συνάρτησης γενικά προκαλεί μια συγκεκριμένη επιβάρυνση (arguments stacking, άλματα, κ.λπ.) και συνεπώς για πολύ σύντομες λειτουργίες μπορεί να είναι πιο αποτελεσματική η απλή εισαγωγή του κώδικα της συνάρτησης όπου ονομάζεται, αντί να εκτελέσετε τη διαδικασία της επίσημης κλήσης μιας συνάρτησης.
Πριν από μια δήλωση λειτουργίας με τον προσδιοριστή inline ενημερώνει τον μεταγλωττιστή ότι η γραμμική επέκταση προτιμάται έναντι του συνήθους μηχανισμού κλήσης λειτουργίας για μια συγκεκριμένη λειτουργία. Αυτό δεν αλλάζει καθόλου τη συμπεριφορά μιας συνάρτησης, αλλά απλώς χρησιμοποιείται για να προτείνει στον μεταγλωττιστή ότι ο κώδικας που παράγεται από το σώμα λειτουργίας θα εισαχθεί σε κάθε σημείο που καλείται η συνάρτηση, αντί να καλείται με μια κανονική κλήση λειτουργίας.
Για παράδειγμα, η συνάρτηση concatenate παραπάνω μπορεί να δηλωθεί ως:

```
inline συμβολοσειρά συμβολοσειράς (const string & a, const string & b)
{
επιστροφή a + b;
}}
```

Αυτό ενημερώνει τον μεταγλωττιστή ότι όταν ονομάζεται concatenate, το πρόγραμμα προτιμά τη λειτουργία να επεκταθεί εν σειρά, αντί να εκτελεί τακτική κλήση. inline ορίζεται μόνο στη δήλωση λειτουργίας, όχι όταν καλείται.
Σημειώστε ότι οι περισσότεροι μεταγλωττιστές βελτιστοποιούν ήδη τον κώδικα για να παράγουν ενσωματωμένες λειτουργίες όταν βλέπουν την ευκαιρία να βελτιώσουν την αποδοτικότητα, ακόμα και αν δεν σημειώνονται ρητά με τον προσδιοριστή inline. Επομένως, αυτός ο προσδιοριστής δηλώνει απλώς ότι ο μεταγλωττιστής που είναι γραμμικός είναι προτιμώμενος για αυτή τη λειτουργία, αν και ο μεταγλωττιστής είναι ελεύθερος να μην τον εισάγει και να βελτιστοποιεί αλλιώς. Στην C ++, η βελτιστοποίηση είναι μια εργασία που μεταβιβάζεται στον μεταγλωττιστή, ο οποίος είναι ελεύθερος να παράγει οποιοδήποτε κώδικα για όσο χρονικό διάστημα η προκύπτουσα συμπεριφορά είναι εκείνη που καθορίζεται από τον κώδικα.

40).Προεπιλεγμένες τιμές στις παραμέτρους.

Στην C ++, οι λειτουργίες μπορούν επίσης να έχουν προαιρετικές παραμέτρους, για τις οποίες δεν απαιτούνται επιχειρήματα στην κλήση, έτσι ώστε, για παράδειγμα, μια λειτουργία με τρεις παραμέτρους να μπορεί να καλείται με μόνο δύο. Για αυτό, η συνάρτηση πρέπει να περιλαμβάνει μια προεπιλεγμένη τιμή για την τελευταία παράμετρο της, η οποία χρησιμοποιείται από τη συνάρτηση όταν καλείται με λιγότερα επιχειρήματα. Για παράδειγμα:

```
// προεπιλεγμένες τιμές στις λειτουργίες
#include <iostream>
χρησιμοποιώντας τον χώρο ονομάτων std.

διαιρεί int (int a, int b = 2)
{
int r;
r = a / b;
επιστροφή (r);
```

}}

int main ()
{
cout << διαίρεση (12) << '\ n';
cout << διαίρεση (20,4) << '\ n';
επιστροφή 0?
}}

Επεξεργασία και εκτέλεση

Σε αυτό το παράδειγμα, υπάρχουν δύο κλήσεις για τη λειτουργία διαίρεσης. Εγώ

int protofunction (int first, int second)
int protofunction (int, int);

Τέλος πάντων, συμπεριλαμβανομένου ενός ονόματος για κάθε παράμετρο βελτιώνει πάντα την αναγνωσιμότητα της δήλωσης.

// declaring functions prototypes
#include <iostream>
using namespace std;

void odd (int x);
void even (int x);

int main()
{
int i;
do {
cout << "Please, enter number (0 to exit): ";
cin >> i;
odd (i);
} while (i!=0);
return 0;
}

void odd (int x)
{
if ((x%2)!=0) cout << "It is odd.\n";
else even (x);
}

void even (int x)
{
if ((x%2)==0) cout << "It is even.\n";
else odd (x);
} Please, enter number (0 to exit): 9
It is odd.
Please, enter number (0 to exit): 6
It is even.

Please, enter number (0 to exit): 1030
It is even.
Please, enter number (0 to exit): 0
It is even.

Το παράδειγμα αυτό δεν αποτελεί πράγματι παράδειγμα αποτελεσματικότητας. Μπορείτε να γράψετε τον εαυτό σας μια έκδοση αυτού του προγράμματος με τις μισές γραμμές του κώδικα. Εν πάση περιπτώσει, αυτό το παράδειγμα δείχνει πώς μπορούν να δηλωθούν οι λειτουργίες πριν από τον ορισμό του:

Οι ακόλουθες γραμμές:

άκυρη περιττή (int a);
άκυρη ακόμη (int a);

Δηλώστε το πρωτότυπο των λειτουργιών. Περιέχουν ήδη όλα όσα είναι απαραίτητα για να τα καλέσουν, το όνομά τους, τους τύπους των επιχειρημάτων τους και τον τύπο επιστροφής τους (άκυρη σε αυτή την περίπτωση). Με αυτές τις πρωτότυπες δηλώσεις, μπορούν να κληθούν πριν οριστούν πλήρως, επιτρέποντας, για παράδειγμα, να τοποθετήσετε τη λειτουργία από εκεί που ονομάζονται (κυρίως) πριν από τον ορισμό αυτών των λειτουργιών.

Ωστόσο, η δήλωση λειτουργιών πριν οριστεί δεν είναι μόνο χρήσιμη για την αναδιοργάνωση της σειράς λειτουργιών εντός του κώδικα. Σε ορισμένες περιπτώσεις, όπως σε αυτή τη συγκεκριμένη περίπτωση, απαιτείται τουλάχιστον μία από τις δηλώσεις, διότι το παράξενο και ομοιόμορφο λέγεται. Υπάρχει μια κλήση για ακόμη και σε περίεργο και μια κλήση να περίεργο στο ζυγό. Επομένως, δεν υπάρχει τρόπος να δομηθεί ο κώδικας, έτσι ώστε να οριστεί το περίεργο πριν από το ζυγό, ακόμα και πριν από το περίεργο.

41).Αναδρομικότητα.

Αναδρομικότητα είναι η ιδιότητα που οι λειτουργίες πρέπει να ονομάζονται από τους ίδιους. Είναι χρήσιμο για ορισμένες εργασίες, όπως στοιχεία διαλογής ή για τον υπολογισμό του παράγοντα των αριθμών. Για παράδειγμα, για να αποκτήσουμε τον παράγοντα ενός αριθμού (n!) Ο μαθηματικός τύπος θα είναι:

n! = n * (η-1) * (η-2) * (η-3) ... * 1
Πιο συγκεκριμένα, 5! (συντελεστής 5) θα είναι:

5! = 5 * 4 * 3 * 2 * 1 = 120
Και μια αναδρομική λειτουργία για να υπολογίσει αυτό σε C + + θα μπορούσε να είναι:

#factorial calculator
#include <iostream>
χρησιμοποιώντας τον χώρο ονομάτων std.

μακρύ παράγοντα (μήκος a)
{
εάν (a> 1)
επιστροφή (a * factorial (a-1))?
αλλού
επιστροφή 1;

```
}}

int main ()
{
μακρύς αριθμός = 9;
cout << αριθμός << "! =" << factorial (αριθμός);
επιστροφή 0?
}}
```

9! = 362880

Επεξεργασία και εκτέλεση

Παρατηρήστε πως σε συνάρτηση factorial συμπεριλάβαμε μια κλήση προς τον εαυτό της, αλλά μόνο αν το επιχείρημα που πέρασε ήταν μεγαλύτερο από 1, δεδομένου ότι, διαφορετικά, η λειτουργία θα εκτελέσει έναν άπειρο αναδρομικό βρόχο, ο οποίος μόλις φθάσει στο 0, θα συνεχίσει να πολλαπλασιάζεται από όλους τους αρνητικούς αριθμούς (πιθανώς προκαλώντας υπερχείλιση στοίβου σε κάποιο σημείο κατά τη διάρκεια του χρόνου εκτέλεσης).

Επεξήγηση γλώσσας C++

ΚαραλήςΓιώργος