C++600

Γενική Περιγραφή

Η γλώσσα C++600 μοιάζει με τη γλώσσα υψηλού επιπέδου C++, και ορίζεται στη συνέχεια. Η C++600 είναι δομημένη με σύνθετες εντολές, οι οποίες είναι υποσύνολο των εντολών της κλασικής C++. Σε σχέση με την κλασική C, στην C++600 δεν περιλαμβάνονται τύποι δεικτών και δομών, περιλαμβάνεται όμως ένας τύπος λίστας που θυμίζει την αντίστοιχη δομή της LISP, καθώς και ένας τύπος ορμαθού χαρακτήρων, ενώ υποστηρίζονται στατικές μεταβλητές. Η C++600 υποστηρίζει κλάσεις και τα βασικά στοιχεία αντικειμενοστραφούς προγραμματισμού της C++, όπως ορισμούς πεδίων και μεθόδων σε τρία επίπεδα προστασίας αναφοράς, απλή κληρονομικότητα, αλλά όχι πολυμορφισμούς ή πολλαπλή κληρονομικότητα. Τέλος, η C++600 επιτρέπει τον ορισμό υποπρογραμμάτων σε ένα μόνο – το εξωτερικό – επίπεδο, όπως η C, αλλά διατηρεί εμβέλειες μεθόδων σε κλάσεις, όπως στις μεθόδους της C++.

Ειδική Περιγραφή

Α. Λεκτικές Μονάδες

Οι λεκτικές μονάδες που αποτελούν και τα τερματικά σύμβολα της γραμματικής της C++600 περιγράφονται στη συνέχεια. Σε παρένθεση – όπου χρειάζεται – δίνονται τα αντίστοιχα συμβολικά ονόματα που εμφανίζονται στη γραμματική της C++600.
Μαζί με τις λεκτικές μονάδες δίνεται και η περιγραφή των σχολίων της C++600, τα οποία όμως δεν εμφανίζονται στη γραμματική της γλώσσας.
Σημειώνεται ότι στη γλώσσα C++600 δεν υπάρχει διάκριση μεταξύ κεφαλαίων και πεζών αλφαβητικών χαρακτήρων, εκτός αν αυτοί αποτελούν μέρος της λέξης μιας λεκτικής μονάδας `CCONST` ή `SCONST`.

Λέξεις-κλειδιά

Οι παρακάτω λέξεις που αποτελούν ανεξάρτητες λεκτικές μονάδες της C++600:
`TYPEDEF CHAR INT FLOAT STRING CLASS PRIVATE PROTECTED PUBLIC VOID STATIC UNION ENUM LIST CONTINUE BREAK IF ELSE WHILE FOR SWITCH CASE DEFAULT RETURN LENGTH NEW CIN COUT MAIN THIS`
Άλλες λέξεις-κλειδιά δίνονται πιο κάτω ως τελεστές.

Αναγνωριστικά (`ID`)

Συμβολοσειρές που αρχίζουν με προαιρετικό χαρακτήρα '_', ακολουθούμενο από αλφαβητικό χαρακτήρα, ακολουθούμενο από μηδέν ή περισσότερους αλφαριθμητικούς χαρακτήρες ή χαρακτήρες '_', και δεν αποτελούν λέξεις-κλειδιά. Δεν επιτρέπονται διαδοχικοί χαρακτήρες '_', εκτός από την αρχή του αναγνωριστικού.
Αποδεκτά παραδείγματα:

```
a100_version_2
__a100_version2
a100_version2_
```

Μη αποδεκτά παραδείγματα[1]:

```
a100__version2
100_version_2
a100_version2__
a100--version-2
```

[1] Σε όλα τα μη αποδεκτά παραδείγματα που δίνονται, η συμβολοσειρά δεν αναγνωρίζεται συνολικά, είναι όμως δυνατό να αναγνωρίζονται μέρη αυτής ως ανεξάρτητες λεκτικές μονάδες.

Απλές σταθερές

Μη προσημασμένες ακέραιες (ICONST):
Ο μοναδικός χαρακτήρας ‘0’, που παριστάνει τη σταθερά με τιμή 0. Επίσης, ένας ή περισσότεροι αριθμητικοί χαρακτήρες, που ο πρώτος δεν είναι ο ‘0’, οπότε η τιμή που παριστάνεται είναι ο αντίστοιχος αριθμός σε δεκαδική βάση. Επίσης, η συμβολοσειρά “0X” ακολουθούμενη από έναν ή περισσότερους αριθμητικούς χαρακτήρες που ο πρώτος δεν είναι ο ‘0’, ή από έναν ή περισσότερους από τους αλφαβητικούς χαρακτήρες ‘A’, ‘B’, ‘C’, ‘D’, ‘E’ και ‘F’. Στην περίπτωση αυτή, η τιμή που παριστάνεται είναι ο αντίστοιχος αριθμός – μετά το πρόθεμα “0X” – σε δεκαεξαδική βάση. Ακόμα, η συμβολοσειρά “0O” ακολουθούμενη από έναν ή περισσότερους αριθμητικούς χαρακτήρες εκτός των ‘8’ και ‘9’ που ο πρώτος δεν είναι ο ‘0’. Στην περίπτωση αυτή, η τιμή που παριστάνεται είναι ο αντίστοιχος αριθμός – μετά το πρόθεμα “0O” – σε οκταδική βάση. Τέλος, η συμβολοσειρά “0B” ακολουθούμενη από έναν ή περισσότερους από τους αριθμητικούς χαρακτήρες ‘0’ και ‘1’ που ο πρώτος δεν είναι ο ‘0’, οπότε η τιμή που παριστάνεται είναι ο αντίστοιχος αριθμός – μετά το πρόθεμα “0B” – σε δυαδική βάση.
Αποδεκτά παραδείγματα:

```
0
180
0X9F0
0o57
0B1001
```

Μη αποδεκτά παραδείγματα:

```
0180
XB7
0X0
0O91
0b010
0XG8A
```

Μη προσημασμένες πραγματικές (FCONST):
Μηδέν ή περισσότεροι αριθμητικοί χαρακτήρες που ακολουθούνται από το χαρακτήρα ‘.’ και τουλάχιστον έναν αριθμητικό χαρακτήρα. Ακολουθεί προαιρετικά πεδίο εκθέτη, που ξεκινάει με τον χαρακτήρα ‘E’, ακολουθούμενο από προαιρετικό πρόσημο και τουλάχιστον έναν αριθμητικό χαρακτήρα. Εναλλακτικά, μη προσημασμένη ακέραια σταθερά που ακολουθείται υποχρεωτικά από πεδίο εκθέτη. Αν δεν υπάρχει πεδίο εκθέτη, ο αριθμός μπορεί να είναι σε δεκαεξαδική βάση, με πρόθεμα “0X”, σε οκταδική βάση, με πρόθεμα “0O”, ή σε δυαδική βάση, με πρόθεμα “0B”. Το ακέραιο μέρος μιας πραγματικής σταθεράς, όπως και το αριθμητικό μέρος του εκθέτη, δε μπορεί να ξεκινά με ‘0’ αν δεν είναι “0”. Σε κάθε περίπτωση που υπάρχει κλασματικό μέρος, αυτό πρέπει να περιλαμβάνει τουλάχιστον ένα χαρακτήρα διαφορετικό από τον ‘0’, εκτός αν είναι “0”.
Αποδεκτά παραδείγματα:

```
180E-2
.5
180.100
7.0
0XA.9
0B1.0010
0O3.617
0x.00B9CF
```

Μη αποδεκτά παραδείγματα:

```
180E-2.2
.E-2
1100
5.
7.00
.5G-2
```

```
05.2E-05
0XBE-2
0B01.1
```

Χαρακτήρες (CCONST):
Οποιοσδήποτε εκτυπώσιμος ASCII χαρακτήρας (κωδικοί 32-126) μεταξύ δύο εμφανίσεων του ειδικού χαρακτήρα ‘'’. Επιπλέον, ειδικοί χαρακτήρες ASCII παριστάνονται με τη βοήθεια του χαρακτήρα ‘\’. Πιο συγκεκριμένα, ο χαρακτήρας LF (Line Feed) παριστάνεται ως `'\n'`, ο χαρακτήρας FF (Form Feed) ως `'\f'`, ο χαρακτήρας HT (Horizontal Tab) ως `'\t'`, ο χαρακτήρας CR (Carriage Return) ως `'\r'`, ο χαρακτήρας BS (BackSpace) ως `'\b'` και ο χαρακτήρας VT (Vertical Tab) ως `'\v'`.
Αποδεκτά παραδείγματα:

```
'a'
'$'
' '
'''
'\n'
'\'
```

Μη αποδεκτά παραδείγματα:

```
'ac'
'\p'
'\\'
```

Ορμαθοί χαρακτήρων (SCONST):
Οποιαδήποτε συμβολοσειρά μεταξύ δύο εμφανίσεων του ειδικού χαρακτήρα ‘"’, συμπεριλαμβανομένης της κενής συμβολοσειράς. Ο χαρακτήρας ‘"’ και οι πιο πάνω ειδικοί χαρακτήρες ASCII παριστάνονται σε έναν ορμαθό χαρακτήρων με τη βοήθεια του χαρακτήρα ‘\’. Με κάθε άλλη χρήση του χαρακτήρα ‘\’ παριστάνεται ο χαρακτήρας που ακολουθεί. Έτσι, ο χαρακτήρας ‘\’ καθαυτός παριστάνεται ως `\\`. Ειδικά όταν ο χαρακτήρας ‘\’ βρίσκεται στο τέλος της γραμμής, ο ορμαθός συνεχίζεται στην επόμενη γραμμή, χωρίς οι χαρακτήρες ‘\’ και αλλαγής γραμμής να αποτελούν μέρος αυτού.
Αποδεκτά παραδείγματα:

```
"CHARACTER +"
""
"STRINGS START AND END WITH \""
"CHARACTER \\ AT THE END OF THE LINE \
 EXTENDS STRING IN THE NEXT LINE\n"
```

Τελεστές

Λογικό Η (OROP): `||`
Λογικό ΚΑΙ (ANDOP): `&&`
Τελεστές σύγκρισης ισότητας (EQUOP): `== !=`
Τελεστές σύγκρισης ανισότητας (RELOP): `> >= < <=`
Προσθετικοί τελεστές (ADDOP): `+ -`
Πολλαπλασιαστικοί τελεστές (MULOP): `* / %`
Λογικό ΟΧΙ (NOTOP): `!`
Τελεστές αυξομείωσης (INCDEC): `++ --`
Τελεστής μεγέθους τύπου (SIZEOP): SIZEOF

Στοιχεία λίστας (LISTFUNC)

Ένα ‘A’ και οσαδήποτε ‘D’, ή κανένα ‘A’ και τουλάχιστον ένα ‘D’, μεταξύ ‘C’ και ‘R’.
Αποδεκτά παραδείγματα:

```
CAR
CDR
```

```
CADDR
```

Μη αποδεκτά παραδείγματα:

```
CARD
CDAR
```

Άλλες λεκτικές μονάδες

Άλλοι χαρακτήρες ή ακολουθίες χαρακτήρων που αποτελούν ανεξάρτητες λεκτικές μονάδες είναι:
‘(’ (LPAREN), ‘)’ (RPAREN), ‘;’ (SEMI), ‘.’ (DOT), ‘,’ (COMMA), ‘=’ (ASSIGN), ‘:’ (COLON), ‘[’ (LBRACK), ‘]’ (RBRACK), ‘&’ (REFER), ‘{’ (LBRACE), ‘}’ (RBRACE), “::” (METH), “>>” (INP), “<<” (OUT), (EOF)
Σε ορισμένες περιπτώσεις κάποιες από τις παραπάνω λεκτικές μονάδες ενεργούν ως τελεστές, όπως αυτό θα φανεί στη γραμματική και σημασιολογία της C++600.
Η λεκτική μονάδα EOF – που ως χαρακτήρας ορίζεται με ειδικό τρόπο από το κάθε σύστημα – δεν εμφανίζεται στη γραμματική της C++600, αλλά πρέπει να παράγεται από το λεκτικό αναλυτή με τιμή 0 για τον τερματισμό της συντακτικής ανάλυσης.

Σχόλια

Τα σχόλια στην C++600 είναι συμβολοσειρές που είτε περικλείονται σε “/*” και “*/” είτε ξεκινούν με “//” και εκτείνονται μέχρι το τέλος της τρέχουσας γραμμής.

Β. Συντακτικοί Κανόνες

Η γραμματική της C++600 περιγράφεται από τους πιο κάτω κανόνες:

```
program → global_declarations main_function

global_declarations → global_declarations global_declaration
      | ε

global_declaration → typedef_declaration
      | enum_declaration
      | class_declaration
      | union_declaration
      | global_var_declaration
      | func_declaration

typedef_declaration → TYPEDEF typename listspec ID dims SEMI

typename → standard_type
      | ID

standard_type → CHAR | INT | FLOAT | STRING | VOID

listspec → LIST | ε

dims → dims dim
      | ε

dim → LBRACK ICONST RBRACK | LBRACK RBRACK

enum_declaration → ENUM ID enum_body SEMI

enum_body → LBRACE id_list RBRACE
```

```
id_list → id_list COMMA ID initializer
      | ID initializer

initializer → ASSIGN init_value
      | ε

init_value → expression
      | LBRACE init_values RBRACE

expression → expression OROP expression
      | expression ANDOP expression
      | expression EQUOP expression
      | expression RELOP expression
      | expression ADDOP expression
      | expression MULOP expression
      | NOTOP expression
      | ADDOP expression
      | SIZEOP expression
      | INCDEC variable
      | variable INCDEC
      | variable
      | variable LPAREN expression_list RPAREN
      | LENGTH LPAREN general_expression RPAREN
      | NEW LPAREN general_expression RPAREN
      | constant
      | LPAREN general_expression RPAREN
      | LPAREN standard_type RPAREN
      | listexpression

variable → variable LBRACK general_expression RBRACK
      | variable DOT ID
      | LISTFUNC LPAREN general_expression RPAREN
      | decltype ID
      | THIS

general_expression → general_expression COMMA general_expression
      | assignment

assignment → variable ASSIGN assignment
      | expression

expression_list → general_expression
      | ε

constant → CCONST | ICONST | FCONST | SCONST

listexpression → LBRACK expression_list RBRACK

init_values → init_values COMMA init_value
      | init_value

class_declaration → CLASS ID class_body SEMI

class_body → parent LBRACE members_methods RBRACE

parent → COLON ID | ε

members_methods → members_methods access member_or_method
      | access member_or_method
```

```
access → PRIVATE COLON | PROTECTED COLON | PUBLIC COLON | ε

member_or_method → member
      | method

member → var_declaration
      | anonymous_union

var_declaration → typename variabledefs SEMI

variabledefs → variabledefs COMMA variabledef
      | variabledef

variabledef → listspec ID dims

anonymous_union → UNION union_body SEMI

union_body → LBRACE fields RBRACE

fields → fields field
      | field

field → var_declaration

method → short_func_declaration

short_func_declaration → short_par_func_header SEMI
      | nopar_func_header SEMI

short_par_func_header → func_header_start LPAREN parameter_types RPAREN

func_header_start → typename listspec ID

parameter_types → parameter_types COMMA typename pass_list_dims
      | typename pass_list_dims

pass_list_dims → REFER
      | listspec dims

nopar_func_header → func_header_start LPAREN RPAREN

union_declaration → UNION ID union_body SEMI

global_var_declaration → typename init_variabledefs SEMI

init_variabledefs → init_variabledefs COMMA init_variabledef
      | init_variabledef

init_variabledef → variabledef initializer

func_declaration → short_func_declaration
      | full_func_declaration

full_func_declaration → full_par_func_header LBRACE decl_statements RBRACE
      | nopar_class_func_header LBRACE decl_statements RBRACE
      | nopar_func_header LBRACE decl_statements RBRACE

full_par_func_header → class_func_header_start LPAREN parameter_list RPAREN
      | func_header_start LPAREN parameter_list RPAREN
```

```
class_func_header_start → typename listspec func_class ID

func_class → ID METH

parameter_list → parameter_list COMMA typename pass_variabledef
      | typename pass_variabledef

pass_variabledef → variabledef
      | REFER ID

nopar_class_func_header → class_func_header_start LPAREN RPAREN

decl_statements → declarations statements
      | declarations
      | statements
      | ε

declarations → declarations decltype typename variabledefs SEMI
      | decltype typename variabledefs SEMI

decltype → STATIC | ε

statements → statements statement
      | statement

statement → expression_statement
      | if_statement
      | while_statement
      | for_statement
      | switch_statement
      | return_statement
      | io_statement
      | comp_statement
      | CONTINUE SEMI
      | BREAK SEMI
      | SEMI

expression_statement → general_expression SEMI

if_statement → IF LPAREN general_expression RPAREN statement if_tail

if_tail → ELSE statement
      | ε

while_statement → WHILE LPAREN general_expression RPAREN statement

for_statement → FOR LPAREN optexpr SEMI optexpr SEMI optexpr RPAREN statement

optexpr → general_expression
      | ε

switch_statement → SWITCH LPAREN general_expression RPAREN switch_tail

switch_tail → LBRACE decl_cases RBRACE
      | single_casestatement

decl_cases → declarations casestatements
      | declarations
```

```
      | casestatements
      | ε

casestatements → casestatements casestatement
      | casestatement

casestatement → CASE constant COLON casestatement
      | CASE constant COLON statements
      | DEFAULT COLON statements

single_casestatement → CASE constant COLON single_casestatement
      | CASE constant COLON statement

return_statement → RETURN optexpr SEMI

io_statement → CIN INP in_list SEMI
      | COUT OUT out_list SEMI

in_list → in_list INP in_item
      | in_item

in_item → variable

out_list → out_list OUT out_item
      | out_item

out_item → general_expression

comp_statement → LBRACE decl_statements RBRACE

main_function → main_header LBRACE decl_statements RBRACE

main_header → INT MAIN LPAREN RPAREN
```

όπου το σύμβολο '|' διαχωρίζει τα εναλλακτικά δεξιά μέλη των κανόνων και ε είναι η κενή συμβολοσειρά.

Οι παραπάνω κανόνες ορίζουν διφορούμενη γραμματική, που με κατάλληλους μετασχηματισμούς ή βοηθητικές περιγραφές προτεραιότητας και προσεταιριστικότητας τελεστών μπορεί να γίνει μη διφορούμενη.

Αρχικό σύμβολο της γραμματικής της C++600 αποτελεί το "program".

Γ. Σημασιολογία

Η σημασιολογία της C++600 καθορίζεται από μια σειρά κανόνων που αφορούν τη δομή ενός *ορθού* προγράμματος. Κάποιοι από τους κανόνες αυτούς είναι πιθανό να καλύπτονται από τη σύνταξη της γλώσσας, ενώ κάποιοι άλλοι μπορούν να χρησιμοποιηθούν για τη μετατροπή της γραμματικής της C++600 σε μη διφορούμενη. Όλοι οι υπόλοιποι κανόνες κωδικοποιούνται στο μεταγλωττιστή της C++600 με τη βοήθεια σημασιολογικών ρουτινών που εκτελούνται κατά τη μετατροπή ενός αρχικού προγράμματος σε ενδιάμεσο κώδικα.

Τύποι δεδομένων

Η C++600 υποστηρίζει τρεις βασικούς τύπους δεδομένων:

- int : ο αριθμητικός τύπος των ακέραιων αριθμών,
- float : ο αριθμητικός τύπος των πραγματικών αριθμών, και

- char : ο τύπος των χαρακτήρων.

Το μέγεθος και η αναπαράσταση των δεδομένων καθενός από τους βασικούς αριθμητικούς τύπους της C++600 καθορίζονται από την τελική γλώσσα μεταγλώττισης. Αυτή επιτρέπει μεγέθη τύπων και αναπαραστάσεις ανάλογα με την υποστήριξη που δίνει η αρχιτεκτονική για τον καθένα.
Τα δεδομένα του τύπου char καταλαμβάνουν τον ελάχιστο χώρο αποθήκευσης που επιτρέπει η τελική γλώσσα, συνήθως ένα byte. Η αναπαράστασή τους γίνεται με τον κώδικα ASCII.
Η ευθυγράμμιση που απαιτείται για την προσπέλαση δεδομένων καθενός από τους πιο πάνω τύπους της C++600 καθορίζεται από την τελική γλώσσα.

Ειδικός βασικός τύπος μπορεί να θεωρηθεί και ο εικονικός τύπος void, ο οποίος δεν αποδίδεται σε δεδομένα, παρά μόνο σε αποτελέσματα συναρτήσεων, όπως θα δούμε παρακάτω.

Εκτός από τους πιο πάνω βασικούς, η C++600 υποστηρίζει και έξι ακόμα τύπους.

Ο τύπος απαρίθμησης είναι ένας βαθμωτός τύπος, του οποίου οι διακριτές τιμές ορίζονται μέσα από μια λίστα αναγνωριστικών. Οι τιμές αυτές θεωρούνται διατεταγμένες από τη μικρότερη προς τη μεγαλύτερη, με τη σειρά εμφάνισής τους στη λίστα, και χρησιμοποιούνται στο πρόγραμμα ως σταθερές.
Ένα παράδειγμα τύπου απαρίθμησης είναι το πιο κάτω:

```
{monday, tuesday, wednesday, thursday, friday}
```

Οι τιμές ενός τύπου απαρίθμησης κωδικοποιούνται ως ακέραιες, με αντίστοιχο μέγεθος και ευθυγράμμιση στην αποθήκευσή τους. Διαδοχικές τιμές του τύπου απαρίθμησης κωδικοποιούνται ως διαδοχικοί ακέραιοι, με την πρώτη τιμή να κωδικοποιείται ως 0.

Ο τύπος string είναι ο τύπος των ορμαθών χαρακτήρων. Συμμετέχει στη γραμματική της C++600 ως βασικός, γι' αυτό και στη συνέχεια θα συμπεριλαμβάνεται σε αυτούς. Σε αντίθεση όμως με τους πιο πάνω βασικούς τύπους, ο τύπος string παράγεται από τον τύπο char, καθώς οι τιμές του προκύπτουν σαν ένα διάνυσμα τιμών τύπου char, με την τελευταία να είναι πάντα 0. Ο τύπος string είναι διατεταγμένος με την λεξικογραφική σειρά των ορμαθών χαρακτήρων που παριστάνονται, με βάση τον κώδικα ASCII των επιμέρους χαρακτήρων. Η αποθήκευση των χαρακτήρων ενός ορμαθού γίνεται σε συνεχόμενες θέσεις μνήμης, μέσα σε χώρο μεγέθους 256 φορές το μέγεθος του τύπου char. Αυτό σημαίνει ότι το μήκος ενός ορμαθού – συμπεριλαμβανομένου του τερματικού 0 – δε μπορεί να ξεπερνάει το 256.

Η C++600 υποστηρίζει και τέσσερις σύνθετους τύπους, τους τύπους πίνακα και λίστας, με στοιχεία δεδομένα του ίδιου τύπου, τον τύπο κλάσης, με στοιχεία δεδομένα όχι αναγκαστικά του ίδιου τύπου, που λέγονται μέλη της κλάσης, ή και συναρτήσεις, που λέγονται μέθοδοι της κλάσης, και τον τύπο ένωσης, με στοιχεία δεδομένα όχι αναγκαστικά του ίδιου τύπου, που λέγονται πεδία της ένωσης. Οι τύποι δεδομένων των στοιχείων πίνακα, μελών κλάσης και πεδίων ενώσεων δεν είναι απαραίτητο να είναι βασικοί. Οι τύποι δεδομένων των στοιχείων λίστας δε μπορούν να είναι σύνθετοι, αλλά ούτε string.
Ένα στοιχείο πίνακα αναπαριστάται με το όνομα του πίνακα και μέσα σε αγκύλες ένα μη αρνητικό ακέραιο δείκτη που μπορεί να πάρει τιμές από 0 έως τον αριθμό των στοιχείων του πίνακα μείον 1. Πίνακες περισσότερων της μιας διάστασης υποστηρίζονται σα διανύσματα πινάκων μικρότερης διάστασης.
Παράδειγμα αναφοράς σε στοιχείο πίνακα είναι το εξής:

```
C[10]
```

ενώ παραδείγματα αναφορών σε στοιχεία πολυδιάστατων πινάκων είναι:

```
C[103][5]
C[0][5][8]
```

Οι παρακάτω αναφορές σε στοιχεία πίνακα δεν είναι αποδεκτές:

```
C[-2]
C[3.5]
```

Δεικτοδότηση με τη βοήθεια μεταβλητής επιτρέπεται μόνο αν ο τύπος δείκτη που προκύπτει είναι ακέραιος. Έτσι, η αναφορά:

```
C[i][k+2]
```

είναι αποδεκτή, μόνο αν ο τύπος του αναγνωριστικού i και της έκφρασης "k+2" είναι ακέραιος.
Σε αντίθεση με τα στοιχεία πίνακα, αναφορά στα οποία γίνεται με δεικτοδότηση, αναφορά στα μέλη και τις μεθόδους του τύπου κλάσης γίνεται ονομαστικά. Έτσι, ένα μέλος ή μια μέθοδος κλάσης αναπαριστάται με το όνομα του αντικειμένου της κλάσης, ακολουθούμενο από το σύμβολο '.' και το όνομα του μέλους ή της μεθόδου.
Παράδειγμα αναφοράς σε μέλος κλάσης είναι το εξής:

```
D.str
```

Όπως και στις κλάσεις, έτσι και στις ενώσεις, αναφορά στα πεδία τους γίνεται ονομαστικά με τον παραπάνω τρόπο.
Ένα μέλος ή μια μέθοδος κλάσης μπορεί σε κάποιες περιπτώσεις να αναπαρασταθεί και χωρίς το όνομα του αντίστοιχου αντικειμένου, όπως θα αναφερθεί παρακάτω.
Παραδείγματα αναφοράς σε στοιχεία πιο πολύπλοκων σύνθετων τύπων είναι τα πιο κάτω:

```
C[n+1][i].x
A.D.z[0][j]
```

όπου τα στοιχεία του δισδιάστατου πίνακα C είναι τύπου κλάσης ή ένωσης, ενώ το στοιχείο D της κλάσης ή ένωσης A είναι επίσης τύπου κλάσης ή ένωσης, με το στοιχείο z της τελευταίας να είναι δισδιάστατος πίνακας. Γενικά δεν υπάρχει περιορισμός σε συνδυασμούς πινάκων, κλάσεων και ενώσεων.
Παραδείγματα αναφοράς σε μεθόδους κλάσεων θα δοθούν πιο κάτω.
Όσο αφορά τον τύπο λίστας, τέλος, αναφορά στα στοιχεία του γίνεται με τη βοήθεια ειδικών συναρτήσεων στοιχείων λίστας. Οι συναρτήσεις αυτές ορίζονται παρακάτω. Μια ολόκληρη λίστα μπορεί επίσης να παρασταθεί και ως έκφραση λίστας με τη βοήθεια αγκυλών. Για παράδειγμα, μια λίστα ακεραίων μπορεί να είναι η:

```
[1,3,-5,0,2]
```

Η αποθήκευση των στοιχείων ενός πίνακα της C++600 στη μνήμη γίνεται σε συνεχόμενες θέσεις, ανάλογα με την ευθυγράμμιση του τύπου αυτών, με αύξουσα σειρά μεταβολής των δεικτών από την τελευταία διάσταση προς την πρώτη. Έτσι, ένας τρισδιάστατος πίνακας A 3×3×2 στοιχείων, αποθηκεύει τα στοιχεία του με τη σειρά: A[0][0][0], A[0][0][1], A[0][1][0], A[0][1][1], A[0][2][0], A[0][2][1], A[1][0][0], A[1][0][1], A[1][1][0], A[1][1][1], A[1][2][0], A[1][2][1], A[2][0][0], A[2][0][1], A[2][1][0], A[2][1][1], A[2][2][0] και A[2][2][1]. Με τον τρόπο αυτό, ένας δισδιάστατος πίνακας αποθηκεύεται γραμμή-γραμμή, όπως δηλαδή υποδηλώνεται από την αναπαράσταση του δισδιάστατου πίνακα σαν διάνυσμα διανυσμάτων.
Ένας σύνθετος τύπος πίνακα καταλαμβάνει τόσες θέσεις αποθήκευσης, όσος είναι ο αριθμός των στοιχείων του επί το μέγεθος του ευθυγραμμισμένου στοιχείου του.

Για τα αντικείμενα κλάσης από την άλλη μεριά, η αποθήκευση στη μνήμη αφορά μόνο τα μέλη αυτών και γίνεται σε συνεχόμενες θέσεις, ανάλογα με την ευθυγράμμισή τους, και με τη σειρά που αυτά συναντώνται στη δήλωση του τύπου της κλάσης.
Ένας σύνθετος τύπος κλάσης καταλαμβάνει τόσες θέσεις αποθήκευσης, όσο είναι το άθροισμα των θέσεων που καταλαμβάνουν τα ευθυγραμμισμένα μέλη του, λαμβάνοντας υπ' όψη την κληρονομικότητα των κλάσεων που περιγράφεται πιο κάτω.

Αντίθετα με τα αντικείμενα κλάσης, τα δεδομένα τύπου ένωσης αποθηκεύουν τα πεδία τους στην ίδια αρχική διεύθυνση, οπότε ο χώρος που καταλαμβάνεται στη μνήμη θα είναι ίσος με το χώρο του μεγαλύτερου πεδίου της ένωσης.

Η αποθήκευση μιας λίστας της C++600 στη μνήμη γίνεται με δυναμικό τρόπο. Ένα στοιχείο λίστας αποθηκεύεται ως ένα διατεταγμένο ζεύγος (περιεχόμενο, διεύθυνση επόμενου στοιχείου). Μια μεταβλητή τύπου λίστας έχει μέγεθος όσο το μέγεθος μιας διεύθυνσης μνήμης και τιμή τη διεύθυνση του πρώτου στοιχείου της λίστας. Το τελευταίο στοιχείο μιας λίστας

έχει τιμή 0 ως διεύθυνση επόμενου στοιχείου. Μια έκφραση λίστας δημιουργεί μια ακολουθία τέτοιων ζευγών. Οι διευθύνσεις στοιχείων παράγονται από κάποιο μηχανισμό δυναμικής εκχώρησης μνήμης, και ως εκ τούτου διαδοχικά στοιχεία δεν αποθηκεύονται κατ' ανάγκη στη σειρά. Κάθε στοιχείο λίστας καταλαμβάνει τόσες θέσεις μνήμης, όσο είναι το άθροισμα των θέσεων του ευθυγραμμισμένου περιεχομένου του και της ευθυγραμμισμένης διεύθυνσης επόμενου στοιχείου.

Δομή ενός προγράμματος C++600

Ένα πρόγραμμα σε γλώσσα C++600 αποτελείται από δηλώσεις τύπων, απαριθμήσεων, κλάσεων, ενώσεων, καθολικών μεταβλητών και μονάδων συναρτήσεων, ακολουθούμενων από τη δήλωση της κύριας μονάδας του προγράμματος. Οι μονάδες συναρτήσεων δηλώνονται στην εξωτερική (καθολική) εμβέλεια του προγράμματος και έχουν την ίδια δομή με την κύρια μονάδα.

Οι δηλώσεις καθολικής εμβέλειας περιλαμβάνουν:
- Δηλώσεις τύπων: Με αυτές αποδίδονται ονόματα σε σύνθετους τύπους της C++600.
- Δηλώσεις απαριθμήσεων: Αυτές ορίζουν βαθμωτούς τύπους απαρίθμησης.
- Δηλώσεις κλάσεων: Αυτές ορίζουν βασικές κλάσεις ή κλάσεις που παράγονται από άλλες που έχουν δηλωθεί παραπάνω.
- Δηλώσεις ενώσεων: Αυτές ορίζουν τύπους ένωσης.
- Δηλώσεις καθολικών μεταβλητών: Οι μεταβλητές αυτές έχουν εμβέλεια ολόκληρο το πρόγραμμα και μπορούν να δέχονται αρχικοποιήσεις των τιμών τους.
- Δηλώσεις μονάδων συναρτήσεων.

Κάθε δήλωση μονάδας περιλαμβάνει:
- Επικεφαλίδα: Η επικεφαλίδα της μονάδας καθορίζει ανάμεσα στα άλλα τις παραμέτρους της, δηλαδή τον τρόπο επικοινωνίας αυτής με τις άλλες μονάδες του προγράμματος.
- Μια σύνθετη εντολή: Αυτή είναι μια σειρά απλών, δομημένων ή άλλων σύνθετων εντολών μεταξύ των χαρακτήρων '{' και '}'. Δηλώσεις τοπικών μεταβλητών με εμβέλεια τη σύνθετη εντολή μπορούν προαιρετικά να προηγούνται των εντολών. Η σειρά εντολών μπορεί να είναι κενή, όπως φαίνεται και στους συντακτικούς κανόνες της γλώσσας.

Η εκτέλεση του κώδικα μιας μονάδας ξεκινάει με την πρώτη και τερματίζεται με την τελευταία εντολή της σύνθετης εντολής της μονάδας, ή με ειδική εντολή της C++600.

Όλες οι δηλώσεις της C++600, καθολικές και τοπικές, τελειώνουν με το χαρακτήρα ';' ή με μια σύνθετη εντολή.

Επικεφαλίδα

Η επικεφαλίδα της κύριας μονάδας ακολουθεί τη σύνταξη των επικεφαλίδων συναρτήσεων που θα παρουσιαστεί σε επόμενη παράγραφο.
Ειδικότερα, ο τύπος της κύριας μονάδας είναι int, γεγονός που υποδηλώνει ότι η κύρια μονάδα επιστρέφει ακέραια τιμή κατά την εκτέλεσή της, ενώ το όνομά της δίνεται από τη λέξη-κλειδί "main", για να τη διαχωρίσει από τις επικεφαλίδες άλλων συναρτήσεων.

Αν και στην κλασική C η κύρια μονάδα έχει παραμέτρους που καθορίζουν την επικοινωνία αυτής με το λειτουργικό σύστημα, στην C++600 η κύρια μονάδα δεν έχει παραμέτρους.

Δηλώσεις καθολικής εμβέλειας

Με μια τέτοια δήλωση αποδίδεται σε ένα αναγνωριστικό ένας τύπος, μια απαρίθμηση, μια κλάση, μια ένωση, μια μεταβλητή ή μια συνάρτηση του προγράμματος. Κάθε τέτοια δήλωση εισάγει το αναγνωριστικό στον πίνακα συμβόλων, με καθολική εμβέλεια. Ένα αναγνωριστικό

μπορεί να δηλωθεί ξανά, πιθανά με άλλο τρόπο, σε κάποια εσωτερική εμβέλεια, οπότε η νέα δήλωση επισκιάζει την παλιά. Ένα αναγνωριστικό δε μπορεί να δηλωθεί για δεύτερη φορά στην ίδια εμβέλεια.
Αξίζει να σημειωθεί ότι δεν υπάρχει προκαθορισμένη σειρά στις δηλώσεις της C++600.

Δηλώσεις τύπων

Μια δήλωση τύπου στην C++600 αποδίδει σε ένα αναγνωριστικό τη δομή που περιγράφει ένα σύνθετο τύπο πίνακα ή λίστας, ή το όνομα ενός άλλου τύπου, ορίζοντας έτσι ένα συνώνυμο του τελευταίου.

Παραδείγματα δηλώσεων τύπων είναι τα παρακάτω:

```
typedef int AR[100][20];
typedef AR ARR[10];
typedef ARR ARR1;
typedef float list list_of_floats;
typedef list_of_floats array_of_list_of_floats[100];
```

όπου η δεύτερη δήλωση πρέπει να ακολουθεί την πρώτη, η τρίτη τη δεύτερη και η πέμπτη την τέταρτη. Η τρίτη από τις παραπάνω δηλώσεις ορίζει το όνομα ARR1 ως συνώνυμο του τύπου ARR.

Σε μια δήλωση πίνακα ο αριθμός στοιχείων του πίνακα για κάθε διάστασή του δίνεται σε αγκύλες, μετά τον τύπο του στοιχείου του και το αναγνωριστικό που δηλώνεται.
Για δήλωση λίστας αναγράφεται η λέξη-κλειδί “list” πριν το όνομα του αναγνωριστικού που δηλώνεται.
Στην ίδια δήλωση τύπου δεν επιτρέπεται να συνυπάρχουν δήλωση πίνακα και δήλωση λίστας, παρόλο που αυτό δεν αποκλείεται από τη σύνταξη της C++600. Τέτοια περίπτωση δήλωσης αποτελεί σημασιολογικό σφάλμα. Πάντως, δεν απαγορεύεται ένας πίνακας λιστών, αρκεί ο πίνακας να ορίζεται σε διαφορετική δήλωση από τη λίστα, όπως φαίνεται και στην τελευταία από τις παραπάνω δηλώσεις.

Δηλώσεις απαριθμήσεων

Μια δήλωση απαρίθμησης της C++600 αποδίδει σε ένα αναγνωριστικό έναν τύπο απαρίθμησης.

Παράδειγμα δήλωσης απαρίθμησης είναι το ακόλουθο:

```
enum workday {monday=1, tuesday, wednesday, thursday, friday};
```

Πιο συγκεκριμένα:

1. Το όνομα μιας απαρίθμησης μπορεί να χρησιμοποιηθεί μετά τη δήλωση αυτής, σαν το όνομα του τύπου, χωρίς τη λέξη-κλειδί “enum”.
2. Οι σταθερές του τύπου απαρίθμησης δίνονται σε μια λίστα μέσα σε άγκιστρα, και διαχωρίζονται μεταξύ τους με το χαρακτήρα ‘,’.
3. Οι σταθερές του τύπου απαρίθμησης δεν επιτρέπεται να αναγράφονται στη δήλωση παραπάνω από μία φορά.
4. Η κωδικοποίηση των σταθερών ενός τύπου απαρίθμησης γίνεται όπως προαναφέρθηκε, αλλά μπορεί και να καθορίζεται στη δήλωση της απαρίθμησης. Έτσι, κάποια απόδοση τιμής σε μια σταθερά ενός τύπου απαρίθμησης αναγκάζει τη σταθερά να πάρει εκείνη την τιμή, οπότε οι σταθερές που ακολουθούν λαμβάνουν διαδοχικές τιμές, μέχρι κάποια πιθανή επόμενη απόδοση τιμής. Στο παραπάνω παράδειγμα, η σταθερά ‘monday’ κωδικοποιείται ως 1, ενώ οι σταθερές που ακολουθούν κωδικοποιούνται ως 2, 3, 4 και 5. Οι τιμές που πιθανά συμμετέχουν σε μια δήλωση απαρίθμησης πρέπει να δίνονται ως ακέραιες σταθερές μετά από το χαρακτήρα ‘=’.

Μετά από μια δήλωση τύπου απαρίθμησης, τα ονόματα των σταθερών του τύπου μπορούν να χρησιμοποιηθούν ως σταθερές καθολικής εμβέλειας μαζί με το όνομα του τύπου και το διαχωριστικό χαρακτήρα '.'. Για παράδειγμα, η πρώτη σταθερά της παραπάνω δήλωσης θα αναφέρεται ως:

```
workday.monday
```

Κάθε σταθερά τύπου απαρίθμησης μπορεί να ανατεθεί σε μεταβλητή συμβατού τύπου.

Δηλώσεις κλάσεων και ενώσεων

Μια δήλωση κλάσης της C++600 αποδίδει σε ένα αναγνωριστικό τη δομή που περιγράφει μια κλάση. Η κλάση μπορεί να είναι βασική ή να παράγεται από άλλη κλάση. Η δήλωση μιας κλάσης περιλαμβάνει εκτός από το όνομά της, το όνομα της κλάσης που την παράγει – αν δεν είναι βασική, καθώς και μία λίστα μελών και μεθόδων. Μια δήλωση ένωσης αποδίδει σε ένα αναγνωριστικό τη δομή που περιγράφει μια ένωση, και περιλαμβάνει εκτός από το όνομά της, και μία λίστα πεδίων.

Παραδείγματα δηλώσεων κλάσεων και ενώσεων είναι τα παρακάτω:

```
class B {
      private:
            int i[2],j[2];
      protected:
            type_V list vlist;
            int go(int list,int&);
};
class Z {
      float a,b;
};
union L {
      char c[4];
      float a;
};
class C : B {
      private:
            int x,y;
      protected:
            char a[40][10];
      public:
            float see, list seelist, seearray[100];
            Z zz;
            L ll;
            union {
                  int i[2];
                  double w;
            };
            void mine(int,int&,float[]);
};
```

όπου type_V είναι απλός τύπος που έχει δηλωθεί νωρίτερα.

Πιο συγκεκριμένα:

1. Το όνομα μιας κλάσης ή ένωσης μπορεί να χρησιμοποιηθεί μόνο μετά την ολοκλήρωση της δήλωσής της. Αναφορά στο όνομα μιας κλάσης ή μιας ένωσης γίνεται χωρίς τη λέξη-κλειδί "class" ή "union", αντίστοιχα.
2. Η δήλωση της υπερκλάσης μιας κλάσης γίνεται με το όνομα αυτής να ακολουθεί το όνομα της κλάσης και το χαρακτήρα ':'. Η υπερκλάση πρέπει να έχει δηλωθεί σε προηγούμενο σημείο.
3. Μια δήλωση μελών κλάσης ή πεδίων ένωσης γίνεται με αναγραφή των ονομάτων τους μετά από ένα όνομα τύπου, βασικού, απαρίθμησης ή σύνθετου που έχει δηλωθεί νωρίτερα. Ο χαρακτήρας ',' διαχωρίζει διαδοχικά μέλη ή πεδία της ίδιας δήλωσης. Κάθε μέλος

ή πεδίο μπορεί να συμπληρώνεται είτε με δήλωση διαστάσεων που το μετατρέπει σε τύπο πίνακα, είτε με δήλωση λίστας που το μετατρέπει σε τύπο λίστας, με στοιχεία του τύπου που αναγράφεται στη δήλωση.

4. Μια δήλωση μεθόδου κλάσης γίνεται με τη μορφή πρωτοτύπου συνάρτησης, που θα περιγραφεί πιο κάτω.
5. Τα αναγνωριστικά των μελών και μεθόδων μιας κλάσης ή των πεδίων μιας ένωσης θεωρούνται ότι βρίσκονται σε εσωτερική εμβέλεια, που δημιουργείται με τη δήλωση της κλάσης ή ένωσης.
6. Όπως προκύπτει από τη σύνταξη της C++600, μια κλάση πρέπει να έχει τουλάχιστον ένα μέλος ή μια μέθοδο, και μια ένωση πρέπει να έχει τουλάχιστον ένα πεδίο.
7. Μια ένωση μπορεί να είναι φωλιασμένη στα μέλη μιας κλάσης, οπότε είναι ανώνυμη, και τα πεδία της αναφέρονται όπως τα μέλη της κλάσης. Η διαφορά από κανονικά μέλη βρίσκεται μόνο στον τρόπο αποθήκευσης των πεδίων της ένωσης, και όχι στον τρόπο αναφοράς.

Η C++600 υποστηρίζει απλή κληρονομικότητα. Έτσι, τα αντικείμενα μιας κλάσης περιέχουν εκτός από τα στοιχεία που δηλώνονται με τη δήλωση της κλάσης, και τα στοιχεία της υπερκλάσης της. Τα κληρονομημένα αυτά στοιχεία θεωρούνται ότι βρίσκονται στην ίδια εμβέλεια με τα υπόλοιπα στοιχεία, γι' αυτό και ένα όνομα κληρονομημένου στοιχείου δεν επιτρέπεται να δηλώνεται ξανά στην παραγόμενη κλάση. Για ένα αντικείμενο παραγόμενης κλάσης, η αποθήκευση των κληρονομημένων μελών στη μνήμη προηγείται της αποθήκευσης των υπόλοιπων μελών της κλάσης.
Τα στοιχεία που δηλώνονται σε μια κλάση μπορούν να αναφέρονται από αντικείμενα αυτής και κάθε παραγόμενης από αυτήν κλάσης, μέσα από μεθόδους της κλάσης. Οι προαιρετικές δηλώσεις "private", "protected" και "public" αποδίδουν δικαίωμα αναφοράς στοιχείων μιας κλάσης σε μεθόδους που δεν ανήκουν στην κλάση. Καθεμιά από αυτές τις δηλώσεις αναφέρεται σε όλα τα στοιχεία που την ακολουθούν, και μέχρι την επόμενη παρόμοια δήλωση, ή μέχρι το τέλος της δήλωσης της κλάσης. Απουσία τέτοιας δήλωσης ισοδυναμεί με δήλωση "public" στην αρχή της δήλωσης της κλάσης. Έτσι:

- Η δήλωση "private" καθορίζει ότι τα στοιχεία που ακολουθούν αναφέρονται μόνο μέσα από μεθόδους αυτής της κλάσης.
- Η δήλωση "protected" καθορίζει ότι τα στοιχεία που ακολουθούν αναφέρονται μέσα από μεθόδους αυτής ή κάποιας παράγωγης κλάσης.
- Η δήλωση "public" καθορίζει ότι τα στοιχεία που ακολουθούν αναφέρονται μέσα από οποιαδήποτε συνάρτηση του προγράμματος.

Ο τρόπος αναφοράς στα στοιχεία μιας κλάσης περιγράφεται αλλού.

Δηλώσεις μεταβλητών

Μια δήλωση μεταβλητών μοιάζει συντακτικά με δήλωση μελών κλάσης και πεδίων ένωσης, και αποδίδει σε ένα ή περισσότερα αναγνωριστικά καθολικές μεταβλητές ενός βασικού τύπου, ή ενός σύνθετου τύπου που έχει δηλωθεί νωρίτερα. Συμπληρωματικά, η δήλωση μιας μεταβλητής μπορεί να συμπεριλαμβάνει αρχικοποίηση της τιμής της ή των τιμών των στοιχείων της.

Παραδείγματα δηλώσεων μεταβλητών είναι τα παρακάτω:

```
int i,j = 0,k = 1;
int n,l[10]; float x = -1.2e-2, y = 2 * x; complex w[100];
char list c; complex a[] = {{0, 2}, {1, -3}, {-4, 0}};
string str = "string", strs[10] = {"joe", "caroline"};
workday first = workday.monday, list anydays;
```

όπου complex και workday είναι τύποι που πρέπει να έχουν δηλωθεί νωρίτερα, με τη δεύτερη να είναι απαρίθμηση που περιέχει τη σταθερά monday. Όπως και με τις δηλώσεις μελών και πεδίων, κάθε αναγνωριστικό μπορεί να συμπληρώνεται είτε με δήλωση διαστάσεων που το μετατρέπει σε τύπο πίνακα, είτε με δήλωση λίστας που το μετατρέπει σε τύπο λίστας, με

στοιχεία του τύπου που αναγράφεται στη δήλωση. Έτσι, σε κάθε δήλωση μπορούν να συνυπάρχουν αναγνωριστικά βαθμωτών μεταβλητών, μεταβλητών τύπου πίνακα και μεταβλητών τύπου λίστας.

Επίσης:

1. Κάθε δήλωση γίνεται για ένα μόνο τύπο, άσχετα αν κάποια αναγνωριστικά συνοδεύονται από δηλώσεις διαστάσεων ή λίστας. Περισσότερες δηλώσεις για τον ίδιο τύπο μπορούν να ακολουθούν πιο κάτω.
2. Η ίδια μεταβλητή δε μπορεί να συνοδεύεται ταυτόχρονα και από δήλωση διαστάσεων και από δήλωση λίστας.
3. Η αρχικοποίηση μιας βαθμωτής μεταβλητής γίνεται με τοποθέτηση της αρχικής τιμής αμέσως μετά το όνομα της μεταβλητής και το χαρακτήρα '='.
4. Για αρχικοποίηση πίνακα, επιτρέπεται η τοποθέτηση πολλαπλών αρχικών τιμών που αποδίδονται σε διαδοχικά στοιχεία του πίνακα – με τη σειρά αποθήκευσης που δόθηκε παραπάνω. Οι πολλαπλές τιμές μπορούν να δίνονται είτε σε ένα εξωτερικό επίπεδο, είτε σε φωλιασμένες ομάδες, ανάλογες της δομής του πίνακα. Για παράδειγμα, οι παρακάτω αρχικοποιήσεις είναι ισοδύναμες:
   ```
   int x[3][2] = {0, 1, 4, -2, 0, -3};
   int x[3][2] = {{0,1},{4,-2},{0,-3}};
   ```
   Στη δεύτερη από τις παραπάνω περιπτώσεις, η αρχικοποίηση των επιμέρους στοιχείων του x ακολουθεί αναδρομικά τους ίδιους κανόνες με την αρχικοποίηση του x. Για πίνακες με στοιχεία άλλου σύνθετου τύπου που είναι δηλωμένος νωρίτερα, πρέπει να ακολουθείται η δεύτερη από τις πιο πάνω μορφές αρχικοποίησης, όπως φαίνεται και πιο πάνω στην αρχικοποίηση του πίνακα a.
5. Η αρχικοποίηση μπορεί να γίνεται με τη βοήθεια έκφρασης, οπότε πρέπει να ακολουθούνται οι κανόνες που διέπουν τις εκφράσεις, όπως αυτοί περιγράφονται πιο κάτω. Επιπλέον, η έκφραση πρέπει να δίνει σταθερή τιμή, κάτι που θα συμβαίνει αν σε αυτή συμμετέχουν μόνο σταθερές, είτε ως λεκτικές μονάδες σταθερών, είτε ως δηλωμένες σταθερές απαρίθμησης, ορατές στο σημείο της παρούσας δήλωσης. Ακόμα, μπορούν να συμμετέχουν άλλες αρχικοποιημένες μεταβλητές, ορατές στην παρούσα δήλωση, οπότε ως τιμή τους λαμβάνεται η τιμή αρχικοποίησής τους. Τη σταθερή τιμή της έκφρασης πρέπει να προσδιορίζει ο σημασιολογικός αναλυτής και να αποδίδει στη μεταβλητή.
6. Σε αρχικοποίηση μεταβλητής τύπου string, οι διαδοχικοί χαρακτήρες του ορμαθού αποθηκεύονται σε διαδοχικές θέσεις του χώρου 256 θέσεων που έχουν δεσμευτεί για τη μεταβλητή, ξεκινώντας από την πρώτη θέση. Μετά τους χαρακτήρες του ορμαθού, οι υπόλοιπες θέσεις του χώρου μέχρι τις 256 λαμβάνουν τιμή 0. Ο ορμαθός που παρέχεται δεν πρέπει να έχει μήκος μεγαλύτερο από 255 χαρακτήρες. Απόδοση αρχικών τιμών για πίνακες στοιχείων τύπου string γίνεται με διαδοχικούς ορμαθούς χαρακτήρων.
7. Δήλωση πίνακα με αρχικοποίηση των τιμών του μπορεί να γίνει χωρίς να αναφέρεται το μέγεθος της πρώτης διάστασής του. Αυτό μπορεί να προκύψει από τον αριθμό των αρχικών τιμών που δίνονται. Από τα προηγούμενα παραδείγματα φαίνεται ότι ο πίνακας a έχει 3 στοιχεία τύπου complex, αφού η λίστα αρχικοποίησης έχει 3 διανύσματα τιμών.
8. Αρχικοποίηση μπορεί να έχει και μια μεταβλητή τύπου κλάσης, ένα αντικείμενο δηλαδή της κλάσης, με παρόμοιο τρόπο. Η λίστα τότε – που μπορεί να είναι ανομοιογενής – δίνει αρχικές τιμές στα μέλη του αντικειμένου, συμπεριλαμβανομένων των όποιων κληρονομημένων μελών, με τη σειρά αποθήκευσής τους στη μνήμη. Μια μεταβλητή ή μέλος κλάσης τύπου ένωσης δέχεται αρχικοποίηση με μία μόνο τιμή που αποδίδεται στο πρώτο πεδίο της ένωσης.
9. Αν ο αριθμός των αρχικών τιμών που αποδίδονται σε έναν πίνακα ή μία κλάση δεν είναι ο ίδιος με τον αριθμό των βασικών στοιχείων ή μελών που δηλώνονται, τότε: αν είναι μικρότερος από το δεύτερο, τα υπόλοιπα στοιχεία του πίνακα ή μέλη της κλάσης θα έχουν μηδενική αρχική τιμή (0, 0.0 ή '\0'), διαφορετικά έχουμε σημασιολογικό σφάλμα. Για πίνακες στοιχείων σύνθετου τύπου, ο παραπάνω κανόνας ελέγχεται για κάθε φωλιασμένη αρχικοποίηση ξεχωριστά. Έτσι, αν complex είναι τύπος πίνακα, για να είναι σωστή η αρχικοποίηση του a, θα πρέπει ο τύπος αυτός να είναι ορισμένος με τουλάχιστον 2 στοιχεία.

Ειδικότερα, αν περιέχει παραπάνω από 2 στοιχεία, τα υπόλοιπα στοιχεία για καθένα από τα στοιχεία του a θα λαμβάνουν μηδενική αρχική τιμή.
10. Δεν επιτρέπεται αρχικοποίηση μεταβλητών τύπου λίστας.
11. Όλες οι καθολικές μεταβλητές – βασικού ή σύνθετου τύπου – που δεν αρχικοποιούνται με κάποια δήλωση αρχικοποίησης λαμβάνουν μηδενική αρχική τιμή κατά την εκκίνηση της εκτέλεσης του προγράμματος.

Όλες οι καθολικές μεταβλητές της C++600 είναι στατικές. Δεν τοποθετούνται στη στοίβα, αλλά έχουν σταθερές διευθύνσεις έξω από αυτή, έχουν δε διάρκεια ζωής όλη τη διάρκεια εκτέλεσης του προγράμματος. Γι' αυτό και οι αρχικές τιμές που τους αποδίδονται με τον παραπάνω τρόπο αποθηκεύονται κατ' ευθείαν σε κατάλληλο χώρο στον κώδικα που παράγει ο μεταγλωττιστής και δεν αποδίδονται κατά την εκτέλεση του προγράμματος.

Δηλώσεις συναρτήσεων

Οι συναρτήσεις είναι ο μοναδικός τύπος υποπρογραμμάτων της C++600 και δηλώνονται μόνο στην εξωτερική εμβέλεια του προγράμματος. Ο έλεγχος της δήλωσης του ονόματος μιας συνάρτησης είναι παρόμοιος με τον έλεγχο δηλώσεων των τύπων, απαριθμήσεων, κλάσεων, ενώσεων και καθολικών μεταβλητών της C++600.

Κάθε συνάρτηση είναι μια μονάδα, και η δήλωσή της περιέχει ολόκληρη τη δομή της, σύμφωνα με τη δομή μονάδας που περιγράφηκε πιο πάνω. Κατ' εξαίρεση υποστηρίζονται και δηλώσεις με τη μορφή πρωτοτύπων, που περιέχουν μόνο επικεφαλίδα.

Η επικεφαλίδα μιας συνάρτησης της C++600 περιέχει με τη σειρά:
(α) Το όνομα του τύπου της συνάρτησης, δηλαδή της τιμής που επιστρέφεται απ' αυτήν.
(β) Το όνομα της συνάρτησης, που πρέπει να είναι μοναδικό στην εξωτερική εμβέλεια του προγράμματος.
(γ) Τις τυπικές παραμέτρους της και τον τρόπο περάσματος αυτών μέσα σε παρενθέσεις.

Παραδείγματα επικεφαλίδων συναρτήσεων είναι τα εξής:

```
float LL (float list y, char &w)
void C::A(int m, int n[10], x_type x, char z[])
void S()
int list III(float&, int list);
```

Ειδικότερα:
1. Ο τύπος μιας συνάρτησης πρέπει να είναι βασικός εκτός από string, ή ακόμα απαρίθμησης ή λίστας. Στην τελευταία περίπτωση τιμή αποτελέσματος θα είναι η διεύθυνση του πρώτου στοιχείου της λίστας. Αν η συνάρτηση δεν επιστρέφει αποτέλεσμα, ο τύπος της είναι void.
2. Αν η συνάρτηση είναι μέθοδος κάποιας κλάσης, το όνομά της περιλαμβάνει και το όνομα της κλάσης ακολουθούμενο από τη λεκτική μονάδα "::".
3. Η δήλωση των τυπικών παραμέτρων περιλαμβάνει τον τύπο τους, ο οποίος μπορεί να είναι βασικός ή να έχει δηλωθεί νωρίτερα. Σε αντίθεση με δηλώσεις μεταβλητών, ένας τύπος μπορεί να αποδοθεί σε μία μόνο παράμετρο.
4. Το πέρασμα μιας παραμέτρου γίνεται κατ' αναφορά, εάν ο αντίστοιχος τύπος είναι string ή σύνθετος, ή εάν το όνομα του τύπου ακολουθεί ο χαρακτήρας '&', διαφορετικά γίνεται κατ' αξία.
5. Το μέγεθος της πρώτης διάστασης ενός τύπου πίνακα που αποτελεί τυπική παράμετρο συνάρτησης δεν είναι αναγκαίο να δίνεται στη δήλωση. Αυτό επιτρέπεται μόνο επειδή, όπως θα αναφερθεί πιο κάτω, η C++600 δεν έχει ισχυρό σύστημα τύπων.
6. Κάθε τυπική παράμετρος που μεταδίδεται κατ' αξία αποτελεί τοπική μεταβλητή της μονάδας της συνάρτησης και αποθηκεύεται στη στοίβα.

7. Μια συνάρτηση μπορεί να μην έχει παραμέτρους.
8. Το πρωτότυπο συνάρτησης είναι επικεφαλίδα που περιλαμβάνει όλα τα παραπάνω, εκτός από τα ονόματα των τυπικών παραμέτρων.
9. Σε μια δήλωση πρωτοτύπου δεν είναι αναγκαία η δήλωση του μεγέθους για καμία διάσταση ενός τύπου πίνακα που αποτελεί τυπική παράμετρο της δήλωσης.
10. Η δήλωση σε μορφή πρωτοτύπου ακολουθείται από το χαρακτήρα ‘;’, επειδή δεν περιλαμβάνει τη σύνθετη εντολή της μονάδας της συνάρτησης.

Όλες οι δηλώσεις που έχουν γίνει με μορφή πρωτοτύπων – άρα και οι δηλώσεις μεθόδων κλάσεων – πρέπει να επαναληφτούν πλήρεις στη συνέχεια. Ο σημασιολογικός αναλυτής πρέπει να επαληθεύει (α) ότι υπάρχει πλήρης δήλωση μιας δήλωσης πρωτοτύπου, και (β) ότι οι τυπικές παράμετροι που δίνονται στην πλήρη δήλωση είναι ακριβώς του τύπου που δόθηκαν στη δήλωση πρωτοτύπου.

Οι συναρτήσεις μπορούν να δηλώνουν τοπικές μεταβλητές στην αρχή μιας σύνθετης εντολής τους. Οι δηλώσεις αυτές είναι παρόμοιες με τις δηλώσεις καθολικών μεταβλητών, με τη διαφορά ότι δε δέχονται αρχικοποιήσεις.
Εφ’ όσον κάθε σύνθετη εντολή της C++600 ορίζει μια νέα εμβέλεια, μια συνάρτηση μπορεί να έχει φωλιασμένες εμβέλειες. Έτσι, κάθε τοπική μεταβλητή μιας συνάρτησης έχει ορατότητα την εμβέλεια στην οποία δηλώνεται, εκτός από όσες φωλιασμένες εμβέλειες την επισκιάζουν.
Κάθε αναφορά σε μια μεταβλητή που δε δηλώνεται στην παρούσα εμβέλεια επιλύεται στην κοντινότερη προς τα έξω εμβέλεια όπου αυτή δηλώνεται. Αν η μεταβλητή δεν είναι τοπική στην παρούσα μονάδα, η αναφορά πρέπει να επιλύεται στην εξωτερική εμβέλεια του προγράμματος.

Μια τοπική μεταβλητή είναι μεταβλητή στοίβας, εκτός αν στη δήλωσή της υπάρχει η λέξη-κλειδί “static”. Σε μια τέτοια περίπτωση, η μεταβλητή αυτή είναι στατική, δεν τοποθετείται στη στοίβα, αλλά στο χώρο καθολικών δεδομένων του προγράμματος, όπου τοποθετούνται και οι καθολικές μεταβλητές του, έχει δε διάρκεια ζωής όλη τη διάρκεια εκτέλεσης του προγράμματος. Ας σημειωθεί ότι η τοποθέτηση της μεταβλητής στο χώρο καθολικών δεδομένων δεν επηρεάζει την ορατότητά της. Καμία εμβέλεια, εξωτερική αυτής στην οποία δηλώνεται, δε μπορεί να περιέχει αναφορά σε αυτήν. Όλες οι στατικές μεταβλητές ενός προγράμματος αρχικοποιούνται σε μηδενική τιμή πριν την εκκίνηση της εκτέλεσης του προγράμματος.

Μια συνάρτηση μπορεί να κληθεί μέσα στη δήλωσή της. Αυτό, σε συνδυασμό με το γεγονός ότι οι τοπικές μεταβλητές μιας συνάρτησης μπορούν να είναι μεταβλητές στοίβας, επιτρέπει την υποστήριξη αναδρομής στην C++600. Με τη μέθοδο δήλωσης πρωτοτύπων συναρτήσεων μπορεί να υποστηριχτεί και έμμεση αναδρομή.

Μέσα στην εμβέλεια μιας συνάρτησης που είναι μέθοδος κλάσης, τα μέλη και οι μέθοδοι του αντικειμένου της κλάσης που κάλεσε τη συνάρτηση αυτή, μπορούν να αναφέρονται χωρίς όνομα αντικειμένου, εκτός από τα ονόματα μελών ή μεθόδων που είναι επισκιασμένα μέσα στην εμβέλεια αυτή. Το υπονοούμενο αντικείμενο σε μια τέτοια περίπτωση μπορεί να αναφερθεί με τη λέξη-κλειδί “this”.

Εκφράσεις

Στη C++600 η έκφραση αποτελεί τη θεμελιώδη απλή εντολή της γλώσσας. Θεωρείται δε εντολή, επειδή, σε αντίθεση με τις περισσότερες άλλες γλώσσες προγραμματισμού προστακτικού τύπου, μπορεί να έχει παρενέργειες.
Στην ενότητα αυτή θα παρουσιάσουμε την αποτίμηση των εκφράσεων της C++600. Οι παρενέργειες αυτών θα περιγραφούν εκτενέστερα στην ενότητα που παρουσιάζει τις εντολές της γλώσσας.

Οι εκφράσεις της C++600 είναι αριθμητικές, τύπου int ή float, ή εκφράσεις λίστας. Λογικές εκφράσεις αναπαριστώνται ως ακέραιες, με τιμή 0 για λογική τιμή «Ψευδής» και τιμή διάφορη του 0 για λογική τιμή «Αληθής». Η αποτίμηση των εκφράσεων της C++600 γίνεται με βάση συγκεκριμένους κανόνες και με τη βοήθεια των τελεστών της γλώσσας.
Κατ' εξαίρεση, ορίζονται και απλές εκφράσεις τύπου char, string ή απαρίθμησης, οι οποίες μπορούν πάντως να συμμετέχουν σε σύνθετες εκφράσεις με τη βοήθεια συγκεκριμένων τελεστών. Τέλος, ορίζονται και απλές εκφράσεις σύνθετου τύπου, που εμφανίζονται μεμονωμένα σε κλήσεις συναρτήσεων, όπως θα δούμε αργότερα.

Μια έκφραση της C++600 περιλαμβάνει:
- Τιμές αριστερής προσπέλασης (L-values), δηλαδή διευθύνσεις του χώρου δεδομένων του προγράμματος που αντιστοιχούν σε μεταβλητές, στοιχεία πινάκων, στοιχεία λίστας, μέλη αντικειμένων κλάσεων ή πεδία ενώσεων.
- Σταθερές, είτε σαν τιμές που υπάρχουν στον αρχικό κώδικα της μονάδας, είτε σαν τιμές σταθερών απαρίθμησης, ορατών στη μονάδα.
- Τελεστές, που επιτρέπουν πράξεις μεταξύ υποεκφράσεων.
- Κλήσεις συναρτήσεων, που έχουν σαν αποτέλεσμα την αντικατάσταση της συνάρτησης στην έκφραση με την τιμή που αυτή επιστρέφει.

Τα τρία τελευταία αντικείμενα ορίζουν τιμές δεξιάς προσπέλασης (R-values).

Παραδείγματα εκφράσεων της C++600 είναι τα εξής:

```
i
a+1
a[3][i+1][j-1]*this.k*2 + (tu(b,z[2]) / (i-1))*i
a+b[j-i].f[2] > 0 and not x or m >= "maria"
z.i+(w.i/2-2.0)/1.2
[3, x+3, f(4*y)]
```

Τιμές αριστερής προσπέλασης

Κάθε διεύθυνση στο χώρο δεδομένων του προγράμματος ονομάζεται τιμή αριστερής προσπέλασης, επειδή μπορεί να βρεθεί στο αριστερό μέλος μιας ανάθεσης. Τέτοιες τιμές στην C++600 αποτελούν οι μεταβλητές, τα στοιχεία πινάκων, τα στοιχεία λίστας, τα μέλη αντικειμένων κλάσεων και τα πεδία ενώσεων.

Όταν μια τιμή αριστερής προσπέλασης βρεθεί σε μια έκφραση, αποτιμάται και μετατρέπεται σε τιμή δεξιάς προσπέλασης. Η τιμή που αποδίδεται γι' αυτήν είναι η τιμή που περιέχεται στη διεύθυνση που αυτή παριστάνει, δηλαδή η τιμή της μεταβλητής, του στοιχείου πίνακα ή λίστας, του μέλους αντικειμένου κλάσης, ή του πεδίου ένωσης. Απαραίτητη προϋπόθεση για αποτίμηση είναι η προηγούμενη δήλωση της αντίστοιχης μεταβλητής, του αντίστοιχου πίνακα, της αντίστοιχης λίστας, του αντίστοιχου αντικειμένου ή της αντίστοιχης ένωσης στην ίδια εμβέλεια ή σε κάποια εξωτερική της. Σε περίπτωση απουσίας τέτοιας δήλωσης υπάρχει σημασιολογικό σφάλμα. Μεταβλητή σύνθετου τύπου μπορεί να εμφανιστεί χωρίς δείκτες, μέλη ή πεδία σε μια έκφραση, μόνο εάν αποτελεί πραγματική παράμετρο στην κλήση μιας συνάρτησης.

Η αποτίμηση μεταβλητής που έχει δηλωθεί κανονικά σε μια μονάδα γίνεται με ανάγνωση της τιμής της μεταβλητής από το χώρο δεδομένων.
Η αποτίμηση στοιχείου πίνακα που επίσης έχει δηλωθεί κανονικά μπορεί να γίνει μετά την αποτίμηση των δεικτών του. Έτσι, υπολογίζεται η διεύθυνση του ζητούμενου στοιχείου, λαμβάνοντας υπ' όψη την αρχική διεύθυνση του πίνακα στο χώρο δεδομένων της μονάδας ή του προγράμματος, τον τρόπο αποθήκευσης των στοιχείων του πίνακα, τις διαστάσεις και το μέγεθός του σε κάθε διάσταση, καθώς και το μέγεθος του στοιχείου του. Στη συνέχεια μπορεί να αποτιμηθεί το στοιχείο αυτό, με ανάγνωσή του από το χώρο δεδομένων.

Η C++600 δεν έχει ισχυρό σύστημα τύπων, κι επομένως δε γίνεται έλεγχος τιμών των δεικτών ενός στοιχείου πίνακα.
Για ένα αντικείμενο κλάσης που έχει δηλωθεί κανονικά, η αποτίμηση ενός μέλους του μπορεί να γίνει, εφ' όσον αυτό είναι δηλωμένο στην κλάση. Η διεύθυνσή του υπολογίζεται λαμβάνοντας υπ' όψη την αρχική διεύθυνση του αντικειμένου στο χώρο δεδομένων της μονάδας ή του προγράμματος, καθώς και το μέγεθος των μελών που προηγούνται, συμπεριλαμβανομένων των κληρονομημένων μελών. Στη συνέχεια το στοιχείο αυτό αποτιμάται με ανάγνωσή του από το χώρο δεδομένων.
Για ένα πεδίο ένωσης που έχει δηλωθεί κανονικά, τέλος, η αποτίμηση μπορεί να γίνει, εφ' όσον αυτό είναι δηλωμένο στην ένωση, με ανάγνωσή του από το χώρο δεδομένων, κατευθείαν από την αρχική διεύθυνση της ένωσης.
Για μια λίστα που έχει δηλωθεί κανονικά, η αποτίμηση ενός στοιχείου της γίνεται μόνο μέσω των ειδικών συναρτήσεων στοιχείων λίστας, οι οποίες με παράμετρο μια λίστα αναφέρονται στο περιεχόμενο και στη διεύθυνση επόμενου στοιχείου ως εξής:

1. Η συνάρτηση CAR() αναφέρεται στο περιεχόμενο του πρώτου στοιχείου της λίστας.
2. Η συνάρτηση CDR() αναφέρεται στη διεύθυνση του επόμενου στοιχείου της λίστας.
3. Η συνάρτηση CADR() αναφέρεται στο περιεχόμενο του δεύτερου στοιχείου της λίστας.
4. Η συνάρτηση CDDR() αναφέρεται στη διεύθυνση του μεθεπόμενου στοιχείου της λίστας.
5. Όμοια ορίζονται και άλλες συναρτήσεις που ακολουθούν την περιγραφή της λεκτικής μονάδας LISTFUNC που δόθηκε νωρίτερα.

Οι συναρτήσεις αυτές είναι τιμές αριστερής προσπέλασης που μέσα σε εκφράσεις αποτιμώνται και επιστρέφουν τις παραπάνω τιμές.
Η αποτίμηση μεταβλητής που δεν είναι δηλωμένη στην παρούσα μονάδα γίνεται στο χώρο καθολικών δεδομένων του προγράμματος.

Οι τυπικές παράμετροι μιας συνάρτησης είναι τιμές αριστερής προσπέλασης στο δυναμικό χώρο δεδομένων της συνάρτησης στη στοίβα, αν μεταδίδονται κατ' αξία, και στον υπόλοιπο χώρο δεδομένων, αν μεταδίδονται κατ' αναφορά. Στη δεύτερη περίπτωση, και σε κάθε κλήση της συνάρτησης, οι αντίστοιχες πραγματικές παράμετροι πρέπει να είναι τιμές αριστερής προσπέλασης, που δεν αποτιμώνται, αλλά περνάνε στη συνάρτηση με τη διεύθυνσή τους. Η αποτίμηση μιας τυπικής παραμέτρου που μεταδίδεται κατ' αναφορά γίνεται σε εκφράσεις που αυτή εμφανίζεται, με ανάγνωση από τη διεύθυνσή της. Ειδικά αν η τυπική παράμετρος είναι σύνθετου τύπου, οπότε και μεταδίδεται κατ' αναφορά, η προσπέλαση κάποιου στοιχείου της γίνεται όπως αναφέρθηκε παραπάνω.
Η λέξη-κλειδί "this" ορίζει μια υπονοούμενη παράμετρο σε μια μέθοδο κλάσης, που ταυτίζεται με το αντικείμενο που καλεί τη μέθοδο αυτή. Η λέξη αυτή χρησιμοποιείται για προσπέλαση των μελών του αντικειμένου, όταν το όνομά τους έχει επισκιαστεί, ή όταν θέλουμε να περάσουμε το αντικείμενο σαν παράμετρο σε άλλη συνάρτηση. Η τιμή της παραμέτρου "this" χρησιμοποιείται για κάθε επίλυση αναφοράς σε μέλη του αντικειμένου, και δεν επιτρέπεται να αλλάξει με κάποια ανάθεση σε αυτήν.

Σταθερές

Οι σταθερές της C++600 είναι αυτές που αναγνωρίζονται άμεσα από το λεκτικό αναλυτή και περιγράφηκαν στο αντίστοιχο κεφάλαιο, καθώς και σταθερές που έχουν δηλωθεί σε κάποια απαρίθμηση, και είναι ορατές στην παρούσα μονάδα.

Ειδικά όσο αφορά τις πρώτες:
Οι τιμές των αριθμητικών σταθερών προκύπτουν άμεσα με μετατροπή των αντίστοιχων λέξεων σε αριθμητικές τιμές, ενώ οι σταθερές τύπου char και string αποδίδονται με την κωδικοποίηση ASCII.
Οι αριθμητικές σταθερές δεν έχουν πρόσημο, και προσημασμένοι αριθμοί προκύπτουν από εκφράσεις με τη βοήθεια του τελεστή προσήμου ADDOP.

Τελεστές

Τελεστές είναι ειδικά σύμβολα, που εφαρμοζόμενα σε έναν αριθμό εκφράσεων – που ονομάζονται *τελούμενα εισόδου* ή *τελεστέοι*, παράγουν μια νέα έκφραση.
Οι τελεστές της C++600 δίνονται στο σχετικό πίνακα και διακρίνονται σε τελεστές με ένα τελούμενο και τελεστές με δύο τελούμενα εισόδου. Στην πρώτη κατηγορία ανήκουν οι τελεστές του προσήμου, της λογικής άρνησης, της εύρεσης μεγέθους και της αυξομείωσης. Οι τελεστές αυτής της κατηγορίας αναγράφονται πριν το τελούμενό τους, εκτός από τους τελευταίους που μπορούν να αναγραφούν και μετά το τελούμενό τους. Στη δεύτερη κατηγορία ανήκουν οι υπόλοιποι τελεστές αριθμητικών και λογικών πράξεων. Οι τελεστές αυτής της κατηγορίας αναγράφονται ανάμεσα στα τελούμενά τους. Ο τελεστής ανάθεσης ανήκει στη δεύτερη κατηγορία, καθώς η τιμή αριστερής προσπέλασης του αριστερού μέρους της ανάθεσης θεωρείται τελούμενο εισόδου αυτής.
Τελεστές με ένα ή δύο τελούμενα εισόδου μπορούν να θεωρηθούν και τα διαχωριστικά σύμβολα ‘(’, ‘)’, ‘[’, ‘]’, ‘,’ και ‘.’ σε περιπτώσεις που θα αναλυθούν στη συνέχεια.

Πιο συγκεκριμένα:

- Στην απλούστερη περίπτωση, ο τελεστής αναφοράς σε στοιχείο κλάσης ή πεδίο ένωσης ‘.’ έχει σύνταξη:
  <αναγνωριστικό> ‘.’ <αναγνωριστικό>
  Ο τελεστής αυτός έχει δύο τελούμενα εισόδου, από τα οποία το πρώτο είναι το όνομα ενός αντικειμένου της κλάσης ή μιας μεταβλητής ένωσης, και το δεύτερο το όνομα ενός μέλους ή μιας μεθόδου της κλάσης, ή ενός πεδίου της ένωσης. Η αναφορά σε μέλος ή πεδίο αντιμετωπίζεται όπως έχει περιγραφεί παραπάνω. Η αναφορά σε μέθοδο αντικειμένου κλάσης συνδυάζεται με την κλήση της μεθόδου με τον τελεστή ‘( )’ που πρέπει να ακολουθεί.
  Στην πιο γενική περίπτωση, είναι δυνατό το πρώτο όνομα να είναι όνομα μέλους αντικειμένου άλλης κλάσης ή πεδίου άλλης ένωσης – πιθανά δεικτοδοτημένο, όταν το στοιχείο αυτό είναι σύνθετου τύπου, ή ακόμα, το δεύτερο να είναι όνομα τύπου κλάσης ή ένωσης, όταν αυτή είναι μέλος ή πεδίο της πρώτης. Τότε, για την αναφορά στο τελευταίο αναγραφόμενο στοιχείο, και για τον υπολογισμό της διεύθυνσής του, πρέπει να προηγηθούν οι επιλύσεις αναφορών για όλους τους προηγούμενους τύπους, που μπορεί να είναι άλλοι σύνθετοι τύποι. Ένα παράδειγμα αναφοράς σε στοιχείο τέτοιας μορφής δίνεται πιο κάτω.
  Αξίζει να σημειωθεί ότι το σύμβολο ‘.’ που χρησιμοποιείται σε κάποια έκφραση αναφοράς σε σταθερά απαρίθμησης δεν είναι τελεστής, αφού δε δρα σε κανένα από τα ονόματα δεξιά και αριστερά του. Ο μεταγλωττιστής πρέπει να αναγνωρίσει σε τέτοια περίπτωση τη σταθερά απαρίθμησης και να παράσχει την τιμή της στη θέση της έκφρασης.
- Στην απλούστερη περίπτωση, ο τελεστής αναφοράς σε στοιχείο πίνακα ‘[ ]’ έχει σύνταξη:
  <αναγνωριστικό> ‘[’ <έκφραση> ‘]’
  Ο τελεστής αυτός έχει δύο τελούμενα εισόδου, από τα οποία το πρώτο είναι το όνομα του πίνακα, και το δεύτερο μια έκφραση, η τιμή της οποίας αποδίδεται σε δείκτη του στοιχείου πίνακα. Η αναφορά στο στοιχείο πίνακα γίνεται όπως περιγράφεται παραπάνω.
  Στην πιο γενική περίπτωση, ο πίνακας μπορεί να είναι στοιχείο άλλου πίνακα, μέλος κλάσης ή πεδίο ένωσης, αλλά και το στοιχείο μπορεί να είναι επίσης τύπου πίνακα. Τότε, για την αναφορά στο τελευταίο αναγραφόμενο στοιχείο πίνακα και τον υπολογισμό της διεύθυνσής του, πρέπει να προηγηθούν οι επιλύσεις αναφορών για τους προηγούμενους τύπους, που μπορεί να είναι άλλοι σύνθετοι τύποι. Ένα παράδειγμα αναφοράς σε στοιχείο πίνακα τέτοιας μορφής δίνεται πιο κάτω.
  Ακόμα, ως πίνακας μπορεί να είναι και κάποια έκφραση αριστερής προσπέλασης τύπου string, οπότε η έκφραση δείκτη αναφέρεται σε ένα συγκεκριμένο χαρακτήρα του ορμαθού. Η αρίθμηση των χαρακτήρων ενός ορμαθού γίνεται ξεκινώντας με 0 από την αρχή του ορμαθού.

Τελεστής	Περιγραφή	Αριθμός τελούμενων	Προσεταιριστικότητα
'.', '[]', '()', POSTINCDEC	Αναφορά σε στοιχείο κλάσης ή ένωσης, αναφορά σε στοιχείο πίνακα, κλήση συνάρτησης, αύξηση / μείωση κατά 1	2, 2, 2, 1	αριστερή
PREINCDEC, SIZEOP	Αύξηση / Μείωση κατά 1, μέγεθος	1, 1	-
MULOP	Πολλαπλασιασμός, διαίρεση (ακέραια / πραγματική)	2, 2	αριστερή
ADDOP	Πρόσημο, πρόσθεση, αφαίρεση, ένωση λιστών και ορμαθών	1, 2, 2, 2	αριστερή
RELOP	Σχεσιακοί τελεστές	2	αριστερή
EQUOP	Τελεστές ισότητας	2	αριστερή
NOTOP	Λογική άρνηση	1	-
ANDOP	Λογικό γινόμενο	2	αριστερή
OROP	Λογικό άθροισμα	2	αριστερή
'='	Ανάθεση	2	δεξιά
','	Διαδοχική αποτίμηση	2	αριστερή
'[]'	Σύνθεση λίστας	1	-

Τελεστές της C++600 σε φθίνουσα σειρά προτεραιότητας

- Ο τελεστής κλήσης συνάρτησης '( )' έχει σύνταξη:
  <όνομα συνάρτησης> '(' <λίστα εκφράσεων> ')'
  Ο τελεστής αυτός έχει δύο τελούμενα εισόδου, από τα οποία το πρώτο είναι το όνομα της συνάρτησης, και το δεύτερο μια λίστα εκφράσεων, οι τιμές των οποίων αποδίδονται στις πραγματικές παραμέτρους της κλήσης. Η αποτίμηση των εκφράσεων στη λίστα γίνεται από τα αριστερά προς τα δεξιά. Η κλήση συνάρτησης θα επεξηγηθεί παρακάτω.
- Ο τελεστής σύνθεσης λίστας '[ ]' έχει σύνταξη:
  '[' <ακολουθία εκφράσεων> ']'
  Ο τελεστής αυτός έχει σα μοναδικό τελούμενο εισόδου μια ακολουθία εκφράσεων, οι τιμές των οποίων αποδίδονται στα διαδοχικά στοιχεία μιας λίστας. Η αποτίμηση των εκφράσεων στην ακολουθία γίνεται από τα αριστερά προς τα δεξιά. Η ακολουθία εκφράσεων μπορεί να είναι κενή, οπότε δημιουργείται μια κενή λίστα, κάτι που ισοδυναμεί με μηδενική διεύθυνση πρώτου στοιχείου.
  Οι επιμέρους εκφράσεις της λίστας πρέπει να είναι του ίδιου βασικού τύπου ή τύπου απαρίθμησης. Το αποτέλεσμα της σύνθεσης είναι τύπου λίστας, που ο μεταγλωττιστής χρησιμοποιεί είτε σε κάποια πράξη, είτε σε κάποια ανάθεση σε τιμή αριστερής προσπέλασης τύπου λίστας. Σε κάθε περίπτωση, ο τύπος των στοιχείων της λίστας πρέπει να είναι συμβατός με τον άλλο τελεστέο αν πρόκειται για πράξη, ή με την τιμή αριστερής προσπέλασης αν πρόκειται για ανάθεση.
  Για τη σύνθεση της λίστας, ο μεταγλωττιστής παράγει κώδικα, ο οποίος να καλεί κάποια συνάρτηση δυναμικής εκχώρησης μνήμης για καθένα από τα στοιχεία της ακολουθίας, ώστε να κατασκευαστούν τα ζεύγη (περιεχόμενο, διεύθυνση επόμενου στοιχείου) που αποτελούν τη λίστα. Το τελευταίο στοιχείο λαμβάνει το 0 ως διεύθυνση επόμενου στοιχείου. Η τιμή του αποτελέσματος της σύνθεσης θα είναι έτσι η διεύθυνση του πρώτου στοιχείου της λίστας.
- Οι τελεστές αυξομείωσης κατά 1 INCDEC που εμφανίζονται δύο φορές στον παραπάνω πίνακα, ως POSTINCDEC όταν είναι τοποθετημένοι *μετά*, και ως PREINCDEC όταν είναι τοποθετημένοι *πριν* το μοναδικό τελεστέο τους. Ο τελεστέος πρέπει να είναι τιμή αριστερής προσπέλασης τύπου int, και το αποτέλεσμα της εφαρμογής είναι τύπου int με τιμή την τιμή του τελεστέου. Σε περίπτωση τελεστών POSTINCDEC, η αποτίμηση του

τελεστέου γίνεται *πριν*, ενώ σε περίπτωση τελεστών PREINCDEC, η αποτίμηση του τελεστέου γίνεται *μετά* την παρενέργεια που θα περιγραφεί πιο κάτω.

- Ο τελεστής εύρεσης μεγέθους SIZEOP. Ο μοναδικός τελεστέος του τελεστή αυτού μπορεί να είναι οποιουδήποτε τύπου. Ακόμα, μπορεί να είναι όνομα τύπου σε παρενθέσεις, και αυτή είναι η μοναδική περίπτωση που όνομα τύπου μπορεί να εμφανίζεται σε έκφραση της C++600. Το αποτέλεσμα της εφαρμογής αυτού του τελεστή είναι μία σταθερά τύπου int με τιμή το μέγεθος σε αριθμό ψηφιολέξεων του τελεστέου. Η εφαρμογή του τελεστή SIZEOP γίνεται από το μεταγλωττιστή, κι επομένως δεν παράγεται κώδικας γι' αυτόν.
- Οι τελεστές προσήμου ADDOP. Ο μοναδικός τελεστέος των τελεστών αυτών πρέπει να είναι αριθμητικού τύπου και το αποτέλεσμα της εφαρμογής τους είναι του ίδιου τύπου.
- Οι αριθμητικοί τελεστές MULOP και ADDOP. Η συμβατότητα τύπων των τελεστέων αυτών και ο τύπος του αποτελέσματος της εφαρμογής τους καθορίζονται στον επόμενο πίνακα:

A \ B	int	float
int	int	float, όχι '%'
float	float, όχι '%'	float, όχι '%'

Τύπος αποτελέσματος αριθμητικής έκφρασης A op B

Όπως δείχνει ο πίνακας, η μόνη περίπτωση μη συμβατότητας τύπων αφορά τον τελεστή MULOP '%', ο οποίος μπορεί να έχει τελούμενα εισόδου μόνο τύπου int. Σε οποιαδήποτε περίπτωση τα τελούμενα εισόδου είναι διαφορετικού τύπου, είναι απαραίτητο ο τελικός κώδικας να περιέχει μετατροπή του ενός τύπου στον τύπο του αποτελέσματος, η δε πράξη πρέπει να γίνεται για τον τύπο του αποτελέσματος.

Ο τελεστής ADDOP '+' εφαρμόζεται και σε τύπους λίστας και ορμαθών χαρακτήρων, οπότε υλοποιεί την ένωση δύο λιστών ή δύο ορμαθών χαρακτήρων. Η ένωση είναι η μόνο πράξη που επιτρέπεται στους δύο αυτούς τύπους. Ειδικότερα για λίστες, η ένωση επιτρέπεται σε λίστες με στοιχεία του ίδιου τύπου. Το αποτέλεσμα της ένωσης είναι του ίδιου τύπου λίστας, όπου η διεύθυνση επόμενου στοιχείου του τελευταίου στοιχείου της πρώτης λίστας έχει λάβει τιμή τη διεύθυνση πρώτου στοιχείου της δεύτερης λίστας. Για ορμαθούς χαρακτήρων, το αποτέλεσμα είναι ένας νέος ορμαθός που περιέχει τους χαρακτήρες του πρώτου χωρίς το τερματικό 0, ακολουθούμενους από τους χαρακτήρες του δεύτερου ορμαθού.

- Ο τελεστής λογικής άρνησης NOTOP. Ο μοναδικός τελεστέος του τελεστή αυτού πρέπει να είναι τύπου int και το αποτέλεσμα είναι του ίδιου τύπου.
- Οι τελεστές λογικών πράξεων ANDOP και OROP. Τα τελούμενα εισόδου αυτών πρέπει να είναι τύπου int. Το αποτέλεσμα της εφαρμογής τους είναι επίσης τύπου int.
- Οι σχεσιακοί τελεστές RELOP. Τα τελούμενα εισόδου των τελεστών αυτών μπορούν να είναι του ίδιου ή διαφορετικού αριθμητικού τύπου. Στην τελευταία περίπτωση, ο κώδικας πρέπει να μετατρέπει το τελούμενο τύπου int σε float. Μετά από πιθανή μετατροπή, η σύγκριση πρέπει να γίνεται για τον τύπο των τελούμενων. Οι τελεστές αυτοί δέχονται και τελούμενα τύπου char, string και απαρίθμησης. Για τις δύο πρώτες περιπτώσεις η σύγκριση γίνεται με βάση τη διάταξη των κωδικών ASCII. Το αποτέλεσμα της εφαρμογής των σχεσιακών τελεστών είναι τύπου int.
- Οι τελεστές ισότητας EQUOP. Τα τελούμενα εισόδου αυτών μπορούν να είναι του ίδιου βασικού τύπου ή τύπου απαρίθμησης, ή διαφορετικού αριθμητικού τύπου. Στην τελευταία περίπτωση, ο κώδικας πρέπει να μετατρέπει το τελούμενο τύπου int σε float. Μετά από πιθανή μετατροπή, η σύγκριση πρέπει να γίνεται για τον τύπο των τελούμενων. Το αποτέλεσμα της εφαρμογής των τελεστών ισότητας είναι τύπου int.

- Ο τελεστής ανάθεσης '='. Το αριστερό τελούμενο εισόδου του τελεστή αυτού πρέπει να είναι τιμή αριστερής προσπέλασης. Ο τύπος και το αποτέλεσμα της εφαρμογής του τελεστή ανάθεσης είναι ο τύπος και η τιμή του αριστερού τελούμενού του μετά την παρενέργεια της ανάθεσης. Η παρενέργεια της ανάθεσης περιγράφεται στην επόμενη ενότητα, όπου δίνονται και περιορισμοί στον τύπο του δεξιού τελούμενού της.
- Ο τελεστής διαδοχικής αποτίμησης ','. Ο τελεστής αυτός δέχεται σαν τελούμενα εισόδου δύο εκφράσεις, τις οποίες και αποτιμά, πρώτα την αριστερή και μετά τη δεξιά, και επιστρέφει την τιμή της δεξιάς έκφρασης.

Παρά την αναπαράσταση των τύπων char και απαρίθμησης με ακέραιες τιμές, στη C++600 δεν υπάρχει συμβατότητα μεταξύ αριθμητικών και τιμών τύπου char ή απαρίθμησης. Μια έκφραση τύπου char ή απαρίθμησης μπορεί να συμμετάσχει σε κάποια μεγαλύτερη έκφραση μόνο μέσω των σχεσιακών τελεστών και τελεστών ισότητας που αναφέρθηκαν.
Εξαίρεση στα παραπάνω αποτελούν οι τελεστές '[ ]' και '( )' που από τη μία προσφέρουν τη δυνατότητα *έμμεσης* συμμετοχής μιας έκφρασης σε μια μεγαλύτερη, αλλά και από την άλλη μπορούν να δώσουν αποτέλεσμα τύπου που δε μπορούν να δώσουν άλλοι τελεστές.

Κάθε λογική πράξη δίνει την ακέραια τιμή 0 για τη λογική τιμή «Ψευδής» και την ακέραια τιμή 1 για τη λογική τιμή «Αληθής».

Σε κάθε εφαρμογή τελεστή, και όταν συμμετέχουν περισσότερα από ένα τελούμενα εισόδου, η αποτίμηση αυτών γίνεται από αριστερά προς τα δεξιά. Απαιτείται αποτίμηση όλων των τελούμενων, εκτός από τα δεξιά τελούμενα λογικών πράξεων που δεν αποτιμώνται, εάν τα αποτελέσματα των πράξεων προκύπτουν με την αποτίμηση των αριστερών τελούμενων. Μ' άλλα λόγια, η C++600 υποστηρίζει βραχυκύκλωση στις λογικές πράξεις.

Μέσα σε μια έκφραση μπορούν να υπάρχουν πολλοί τελεστές. Η σειρά εφαρμογής αυτών για την αποτίμηση της έκφρασης καθορίζεται από παρενθέσεις ή από κανόνες προτεραιότητας και προσεταιριστικότητας. Η σειρά προτεραιότητας των τελεστών καθορίζεται από τη σειρά που αυτοί αναγράφονται στον πρώτο από τους δύο παραπάνω πίνακες, από μεγαλύτερη προς μικρότερη προτεραιότητα. Η προσεταιριστικότητα των τελεστών, δίνεται στον ίδιο πίνακα. Αν δεν έχει νόημα, η προσεταιριστικότητα δεν είναι ορισμένη και αναγράφεται ως '-'.
Σε μια σύνθετη έκφραση δεν επιτρέπεται διαδοχική εφαρμογή τελεστών προσήμου ADDOP, όπως για παράδειγμα στην έκφραση:

```
x > -+3
```

που δεν είναι αποδεκτή.

Ας θεωρήσουμε για παράδειγμα την έκφραση:

```
i*x-y-j*z < -a % k % 2 && ! b.k[i+2][m].a == 'a'
```

στην οποία κατ' αρχήν υποθέτουμε ότι οι τύποι των τιμών που συμμετέχουν είναι οι προβλεπόμενοι.
Στην έκφραση αυτή ο τελεστής ANDOP '&&' έχει τη μικρότερη προτεραιότητα και δε μπορεί να εφαρμοστεί πριν την αποτίμηση του αριστερού τουλάχιστον τελούμενού του. Επίσης ο σχεσιακός τελεστής RELOP '<' στο αριστερό τελούμενο του προηγούμενου – το οποίο και αποτιμάται πριν από το δεξί – εφαρμόζεται μόνο αφού έχουν αποτιμηθεί και τα δύο τελούμενά του. Στο αριστερό τελούμενο του τελευταίου, οι δύο διαδοχικές αφαιρέσεις που ορίζονται από τον τελεστή ADDOP '-' εφαρμόζονται από αριστερά προς τα δεξιά, λόγω αριστερής προσεταιριστικότητας του τελεστή αυτού.
Ο πρώτος τελεστής που θα εφαρμοστεί στην πιο πάνω έκφραση θα είναι ο πρώτος από αριστερά τελεστής MULOP '*' που έχει μεγαλύτερη προτεραιότητα από τον ADDOP. Στη συνέχεια θα εφαρμοστεί ο πρώτος από αριστερά τελεστής ADDOP '-', ενώ πριν εφαρμοστεί ο δεύτερος από τους δύο διαδοχικούς ADDOP, θα πρέπει να εφαρμοστεί ο δεύτερος τελεστής MULOP '*' που υπάρχει στο δεξί τελούμενο του προηγούμενου. Η εφαρμογή του πρώτου από αριστερά τελεστή MULOP '%' προηγείται του δεύτερου, ενώ ο τελεστής προσήμου

ADDOP '-' εφαρμόζεται μετά τους δύο διαδοχικούς MULOP '%', επιτρέποντας στη συνέχεια την εφαρμογή του τελεστή RELOP '<'.
Έτσι ολοκληρώνεται η αποτίμηση του αριστερού τελεστέου του τελεστή ANDOP. Αν αυτή δώσει τιμή 0, σταματάμε την αποτίμηση της έκφρασης, γιατί αυτή θα έχει τιμή 0. Διαφορετικά, συνεχίζουμε με την αποτίμηση του δεξιού τελεστέου του τελεστή ANDOP.
Στο τελούμενο του τελεστή NOTOP '!' αποτιμάται πρώτα το αριστερό τελούμενο του τελεστή EQUOP '==', το οποίο είναι μέλος αντικειμένου κλάσης ή πεδίο ένωσης, το οποίο είναι στοιχείο πίνακα, ο οποίος είναι επίσης στοιχείο πίνακα, ο οποίος είναι μέλος αντικειμένου κλάσης ή πεδίο ένωσης.
Έτσι, προηγείται ο υπολογισμός της αρχικής διεύθυνσης του πίνακα "b.k" που είναι στοιχείο της μεταβλητής "b", επειδή η εφαρμογή των τελεστών '.' και '[ ]' έχει ίδια προτεραιότητα και αριστερή προσεταιριστικότητα. Στη συνέχεια, και εφ' όσον η εφαρμογή του τελεστή '[ ]' γίνεται με αριστερή προσεταιριστικότητα, ακολουθεί η αποτίμηση της έκφρασης του δείκτη που αναγράφεται πρώτος από αριστερά ("i+2"). Έπειτα υπολογίζεται η αρχική διεύθυνση του στοιχείου πίνακα "b.k[i+2]". Ακολουθεί η αποτίμηση της έκφρασης του δείκτη στη δεύτερη εφαρμογή του '[ ]', και ο υπολογισμός της διεύθυνσης του στοιχείου "b.k[i+2][m]", που είναι τύπου κλάσης ή ένωσης. Τέλος, υπολογίζεται η διεύθυνση του στοιχείου a της προηγούμενης κλάσης ή ένωσης, το οποίο αποτιμάται για να μας δώσει τη ζητούμενη τιμή τύπου char.
Η αποτίμηση του δεύτερου τελούμενου του EQUOP '==' δίνει την τιμή της σταθεράς 'a' τύπου char.
Τελειώνοντας, εφαρμόζεται ο παραπάνω τελεστής EQUOP, ακολουθούμενος από τον NOTOP. Ο τελεστής ANDOP δε χρειάζεται να εφαρμοστεί, αφού ξέρουμε ότι ο αριστερός τελεστέος είχε τιμή 1, οπότε το αποτέλεσμα της εφαρμογής του είναι η τιμή του δεξιού τελεστέου.

Γενικά πρέπει να χρησιμοποιούμε παρενθέσεις σε κάθε περίπτωση που επιθυμούμε παρέμβαση στους πιο πάνω κανόνες, όπως προβλέπει η σύνταξη των εκφράσεων της C++600. Για παράδειγμα, η έκφραση:

```
-(-i+1)*(a-(b-c))
```

(α) επιτρέπει διαδοχική εφαρμογή του ίδιου τελεστή προσήμου, (β) επιβάλλει την αποτίμηση του δεξιού τελούμενου του ADDOP '-' πριν από το αριστερό, και (γ) επιβάλλει την εφαρμογή των τελεστών ADDOP '+' και '-' πριν από την εφαρμογή του MULOP '*'.
Επίσης, παρενθέσεις μπορούν να χρησιμοποιηθούν και για βελτίωση της εμφάνισης μιας έκφρασης, χωρίς αναγκαστικά να επηρεάζουν τους κανόνες εφαρμογής των τελεστών που συμμετέχουν σε αυτήν.

Κλήση συναρτήσεων

Αν 'func' είναι το όνομα μιας συνάρτησης με αποτέλεσμα τύπου 'type', τότε η έκφραση

$$\texttt{func}(e_1, e_2, \ldots, e_n)$$

είναι μια τιμή δεξιάς προσπέλασης τύπου 'type'. Η αποτίμηση αυτής γίνεται με την εκτέλεση του κώδικα της μονάδας της συνάρτησης, με τις εξής προϋποθέσεις:

- Ο αριθμός n των εκφράσεων στις παρενθέσεις – οι πραγματικές παράμετροι – πρέπει να είναι ίσος με τον αριθμό των τυπικών παραμέτρων.
- Ο τύπος κάθε πραγματικής παραμέτρου με πέρασμα κατ' αξία πρέπει να είναι συμβατός με τον τύπο της αντίστοιχης τυπικής παραμέτρου, σύμφωνα με τους κανόνες συμβατότητας για ανάθεση που περιγράφονται πιο κάτω.
- Ο τύπος κάθε πραγματικής παραμέτρου με πέρασμα κατ' αναφορά πρέπει να ταυτίζεται με τον τύπο της αντίστοιχης τυπικής παραμέτρου.

Κατά την κλήση μιας συνάρτησης οι πραγματικές παράμετροι αποτιμώνται από αριστερά προς τα δεξιά και τοποθετούνται στη στοίβα, στο χώρο που δεσμεύεται σα χώρος δεδομένων της συνάρτησης, απ' όπου θα μπορεί να τους χρησιμοποιήσει η μονάδα αυτής. Για μια παράμετρο κατ' αναφορά, στη στοίβα τοποθετείται η αντίστοιχη διεύθυνση. Ειδικά για παράμετρο τύπου string, αν η πραγματική παράμετρος είναι ένας σταθερός ορμαθός χαρακτήρων και όχι

τιμή αριστερής προσπέλασης, ο μεταγλωττιστής δεσμεύει χώρο για μια προσωρινή τοπική μεταβλητή τύπου string στο καλούν περιβάλλον, παράγει κώδικα που να αντιγράφει στο χώρο αυτό το περιεχόμενο του ορμαθού μαζί με το τερματικό 0, και περνάει πια ως πραγματική παράμετρο τη διεύθυνση αυτής της μεταβλητής. Όπως και στις αρχικοποιήσεις, ο ορμαθός που παρέχεται δεν πρέπει να έχει μήκος μεγαλύτερο από 255 χαρακτήρες.
Με την έξοδο από τη συνάρτηση, θα πρέπει η τιμή του αποτελέσματος να βρίσκεται σε προκαθορισμένο σημείο αποθήκευσης, επίσης στη στοίβα. Με κάθε κλήση μιας συνάρτησης, η στοίβα μεταβάλλεται, κι έτσι ο χώρος δεδομένων της συνάρτησης, δηλαδή το *εγγράφημα δραστηριοποίησης* της συνάρτησης, είναι δυναμικός.

Εξαίρεση στα παραπάνω αποτελούν οι προκαθορισμένες συναρτήσεις στοιχείων λίστας, οι οποίες δε λειτουργούν ως συναρτήσεις, αλλά απλά απομονώνουν και επιστρέφουν ένα μέρος της λίστας που δίνεται ως παράμετρος. Το μέρος αυτό μάλιστα επιστρέφεται ως τιμή αριστερής προσπέλασης, όπως ακριβώς ένα στοιχείο πίνακα ή μέλος κλάσης ή πεδίο ένωσης.

Εκτός από τις προκαθορισμένες συναρτήσεις στοιχείων λίστας, ορίζεται και η προκαθορισμένη συνάρτηση LENGTH(), η οποία δέχεται ως παράμετρο μια λίστα και επιστρέφει το πλήθος των στοιχείων της λίστας, καθώς και η προκαθορισμένη συνάρτηση NEW(), η οποία δέχεται ως παράμετρο μια τιμή και επιστρέφει μια λίστα ενός στοιχείου, με στοιχείο την τιμή-παράμετρο, και με διεύθυνση επόμενου στοιχείου ίση με 0.
Η πρώτη από τις δύο παραπάνω συναρτήσεις μπορεί να εφαρμοστεί και σε έναν ορμαθό χαρακτήρων, οπότε επιστρέφει το πλήθος των χαρακτήρων του ορμαθού – χωρίς το τερματικό 0.

Ειδικά για μια μέθοδο κλάσης, υπονοείται σαν παράμετρος η τιμή αριστερής προσπέλασης του αντικειμένου που την καλεί. Η παράμετρος αυτή δε δηλώνεται, αλλά τοποθετείται στη στοίβα, ώστε να χρησιμεύσει για την επίλυση αναφορών σε μέλη και μεθόδους του αντικειμένου μέσα στη σύνθετη εντολή της συνάρτησης. Το όνομα δε της καλούμενης συνάρτησης ορίζεται από το όνομα της αντίστοιχης κλάσης και της μεθόδου της. Για παράδειγμα, η κλήση:

```
x.f(i,j+1,a);
```

όταν το αντικείμενο x είναι τύπου μιας κλάσης C, γίνεται στην πραγματικότητα ως:

```
C::f(x,i,j+1,a);
```

Εντολές

Η C++600 υποστηρίζει απλές και δομημένες εντολές, καθώς και τη σύνθετη εντολή. Μια δομημένη εντολή της C++600 περιέχει μία ή περισσότερες απλές, δομημένες ή σύνθετες εντολές. Μια σύνθετη εντολή μπορεί να είναι κενή, αλλά περιέχει οπωσδήποτε τους χαρακτήρες '{', ';' και '}'. Επίσης, κάθε σύνθετη εντολή έχει τη δική της εμβέλεια, στην οποία μπορούν να οριστούν τοπικές μεταβλητές.

Οι απλές εντολές της C++600 είναι:
- Η εντολή έκφρασης
- Η εντολή επιστροφής από συνάρτηση
- Οι εντολές εισόδου/εξόδου
- Η εντολή continue
- Η εντολή break

Οι δομημένες εντολές της C++600 είναι:
- Η εντολή διακλάδωσης
- Η εντολή βρόχου while
- Η εντολή βρόχου for
- Η εντολή επιλογής switch

Κάθε εντολή της C++600 που δεν τελειώνει με σύνθετη εντολή τερματίζεται με το χαρακτήρα ';'.

Η εντολή έκφρασης

Η εντολή έκφρασης επιστρέφει την τιμή που προκύπτει από την αποτίμησή της, με τον τρόπο που περιγράφηκε νωρίτερα.

Παρενέργειες σε μια έκφραση μπορούμε να έχουμε με έναν από τους πιο κάτω δύο τρόπους:
(α) την εφαρμογή του τελεστή αυξομείωσης κατά 1 INCDEC, που ονομάζουμε έκφραση αυξομείωσης, και
(β) την εφαρμογή του τελεστή ανάθεσης '=', που ονομάζουμε έκφραση ανάθεσης.

Η έκφραση αυξομείωσης εκτελεί την αντίστοιχη πράξη και αποθηκεύει το αποτέλεσμά της πίσω στην τιμή αριστερής προσπέλασης που είναι το τελούμενο εισόδου της, δηλαδή στη διεύθυνση μιας μεταβλητής, ενός στοιχείου πίνακα, ενός μέλους αντικειμένου κλάσης ή ενός πεδίου ένωσης του χώρου δεδομένων του προγράμματος. Στη δεύτερη περίπτωση πρέπει να προηγηθεί η αποτίμηση των εκφράσεων των δεικτών και ο υπολογισμός της τιμής αριστερής προσπέλασης του τελούμενου της πράξης, ενώ στις υπόλοιπες περιπτώσεις πρέπει να προηγηθεί ο υπολογισμός της τιμής αριστερής προσπέλασης του μέλους ή πεδίου.
Παραδείγματα εκφράσεων αυξομείωσης είναι τα εξής:

```
a.x++;
--c[i-1][mm[j++]].x;
```

όπου mm είναι πίνακας ακεραίων, ενώ c είναι πίνακας κλάσεων ή ενώσεων.

Η έκφραση ανάθεσης αποδίδει την τιμή μιας έκφρασης σε μια τιμή αριστερής προσπέλασης. Όπως και για την προηγούμενη έκφραση, της ανάθεσης πρέπει να προηγηθεί η αποτίμηση των εκφράσεων των όποιων δεικτών, και να υπολογιστεί η τιμή αριστερής προσπέλασης, αν το αριστερό τελούμενο της ανάθεσης είναι σύνθετου τύπου.
Η ανάθεση γίνεται με αποθήκευση της τιμής της έκφρασης στη διεύθυνση που παριστάνεται από την τιμή αριστερής προσπέλασης.
Παραδείγματα εκφράσεων ανάθεσης είναι τα εξής:

```
a = w = s > 0.0;
ch = 'c';
c[i-1][mm[j]].x = x*x+1;
il = [1];
CDR(il) = [2] + CDR([5,3,-4]);
```

όπου mm είναι πίνακας ακεραίων, c είναι πίνακας κλάσεων ή ενώσεων, ενώ il είναι λίστα.

Για να μπορεί να γίνει ανάθεση, πρέπει η τιμή της έκφρασης να είναι βασικού τύπου ή τύπου απαρίθμησης, συμβατού με τον τύπο της τιμής αριστερής προσπέλασης. Δύο βασικοί τύποι είναι συμβατοί για ανάθεση, εάν:
(α) ταυτίζονται, ή αλλιώς
(β) είναι αριθμητικοί, οπότε συμβαίνει μετατροπή σε πραγματικό για ανάθεση από τύπο int σε τύπο float, ή αποκοπή του κλασματικού μέρους για ανάθεση από τύπο float σε τύπο int.
Δύο τύποι απαρίθμησης είναι συμβατοί για ανάθεση μόνο αν ταυτίζονται.

Σε μια ανάθεση ενός ορμαθού χαρακτήρων σε μια μεταβλητή τύπου string, οι διαδοχικοί χαρακτήρες του ορμαθού αντιγράφονται σε διαδοχικές θέσεις του χώρου 256 θέσεων που έχουν δεσμευτεί για τη μεταβλητή, ξεκινώντας από την πρώτη θέση. Μετά τους χαρακτήρες του ορμαθού, αποθηκεύεται η τιμή 0, ενώ οι υπόλοιπες θέσεις του χώρου μέχρι τις 256 διατηρούν την όποια προηγούμενη τιμή τους. Ο ορμαθός που παρέχεται ως τιμή δεξιάς προσπέλασης στο δεξί μέλος της ανάθεσης δεν πρέπει να έχει μήκος μεγαλύτερο από 255 χαρακτήρες.

Ανάθεση μεταξύ σύνθετων τύπων επιτρέπεται μόνο για λίστες στοιχείων συμβατού τύπου. Τότε, η διεύθυνση πρώτου στοιχείου της λίστας που παριστάνεται με το δεξί μέλος αποδίδεται στην έκφραση λίστας του αριστερού μέλους.
Μια ανάθεση λίστας ελλοχεύει τον κίνδυνο κάποια στοιχεία λίστας να μείνουν "ξεκρέμαστα", δηλαδή να μην είναι πλέον προσπελάσιμα, κάτι που επιβαρύνει το σύστημα δυναμικής εκχώρησης μνήμης με κίνδυνο ασφυξίας. Για το σκοπό αυτό ο μεταγλωττιστής θα πρέπει να παράγει εδώ επιπλέον κώδικα ανίχνευσης και συλλογής ξεκρέμαστων στοιχείων.

Διαδοχικές αναθέσεις στην ίδια έκφραση επιτρέπονται και εκτελούνται από δεξιά προς τα αριστερά, εφ' όσον ο τελεστής της ανάθεσης έχει δεξιά προσεταιριστικότητα, η οποία είναι αναγκαία, επειδή το αποτέλεσμα μιας ανάθεσης είναι τιμή δεξιάς προσπέλασης και όχι αριστερής.

Ας σημειωθεί ότι οι εκφράσεις που δίνονται πιο πάνω και για τις δύο περιπτώσεις μπορούν να είναι υποεκφράσεις μιας μεγαλύτερης έκφρασης.

Η εντολή επιστροφής από συνάρτηση

Η εντολή αυτή επιστρέφει τη ροή του κώδικα από κάποια συνάρτηση στη μονάδα που την κάλεσε.
Όταν η συνάρτηση στην οποία βρίσκεται δεν είναι τύπου void, η εντολή αυτή δέχεται σαν παράμετρο μια έκφραση, οπότε τοποθετεί την τιμή της έκφρασης στη στοίβα. Ο τύπος της έκφρασης πρέπει να ταυτίζεται με τον τύπο της συνάρτησης.
Αν η συνάρτηση είναι τύπου void, η εντολή αυτή δεν επιτρέπεται να συνοδεύεται από έκφραση.

Παραδείγματα επιστροφής από συνάρτηση είναι τα παρακάτω:

```
return k+2;
return;
```

Οι εντολές εισόδου/εξόδου

Αυτές είναι οι εντολές ανάγνωσης και εγγραφής δεδομένων. Συντάσσονται με το όνομα της εντολής και μια λίστα στοιχείων, που στην περίπτωση ανάγνωσης είναι τιμές αριστερής προσπέλασης, ενώ στην περίπτωση εγγραφής είναι εκφράσεις. Διαδοχικά στοιχεία εισόδου διαχωρίζονται μεταξύ τους και με το όνομα της αντίστοιχης εντολής μέσω της λεκτικής μονάδας ">>", ενώ διαδοχικά στοιχεία εξόδου διαχωρίζονται μεταξύ τους και με το όνομα της αντίστοιχης εντολής με τη λεκτική μονάδα "<<".

Παραδείγματα εντολών εισόδου/εξόδου είναι:

```
cin >>n>>x
cout <<"Temperature: "<<f<<"F, or "<<5/9*(f-32)<<"C."
```

Η είσοδος και η έξοδος γίνονται στα συνήθη αρχεία εισόδου/εξόδου χωρίς προδιαγραφές, δηλαδή προηγούμενο καθορισμό του τύπου και της μορφής των στοιχείων που διαβάζονται ή γράφονται. Με κάθε εντολή εξόδου γράφεται μια γραμμή κειμένου στην έξοδο του προγράμματος.

Η εντολή continue

Η εντολή αυτή επιτρέπεται να βρίσκεται μόνο μέσα στη σύνθετη εντολή μιας δομημένης εντολής βρόχου. Εκτέλεσή της οδηγεί στην επόμενη επανάληψη του βρόχου. Αν ο βρόχος είναι φωλιασμένος σε άλλο βρόχο, η εντολή αναφέρεται στον πιο εσωτερικό βρόχο που την περιέχει στη σύνθετη εντολή του.

Η εντολή break

Η εντολή αυτή επιτρέπεται να βρίσκεται μόνο μέσα στη σύνθετη εντολή μιας δομημένης εντολής βρόχου ή μιας δομημένης εντολής επιλογής. Εκτέλεσή της οδηγεί στον άμεσο τερματισμό του βρόχου ή την έξοδο από την εντολή επιλογής. Αν ο βρόχος ή η εντολή επιλογής είναι φωλιασμένος σε άλλο βρόχο ή εντολή επιλογής, η εντολή αναφέρεται στον πιο εσωτερικό βρόχο ή την πιο εσωτερική εντολή επιλογής που την περιέχει στη σύνθετη εντολή του.

Η εντολή διακλάδωσης

Η εντολή διακλάδωσης if είναι μια δομημένη εντολή με παραμέτρους μια έκφραση σε παρενθέσεις και μία ή δύο άλλες εντολές.

Η εκτέλεση της εντολής διακλάδωσης ξεκινά με την αποτίμηση της έκφρασης. Αν αυτή έχει τιμή που αντιστοιχεί στη λογική τιμή «Αληθής», εκτελείται η εντολή που ακολουθεί την έκφραση, ενώ αν υπάρχει η λέξη-κλειδί “else”, τότε η εντολή που ακολουθεί αυτή τη λέξη δε θα εκτελεστεί. Εάν η έκφραση έχει τιμή που αντιστοιχεί στη λογική τιμή «Ψευδής» και υπάρχει η αντίστοιχη λέξη-κλειδί “else”, τότε εκτελείται μόνο η εντολή που ακολουθεί αυτή τη λέξη, ενώ αν δε υπάρχει η λέξη-κλειδί “else”, δεν εκτελείται καμία εντολή.

Ένα παράδειγμα εντολής διακλάδωσης είναι το παρακάτω:

```
if (a[i+1] > 13.5 and z[j]) {
      a[i-1] = f(z[j+1],i-1)*2;
      if (x.play == 0) b.over = a[i];
} else
      a[i-1] = f(z[n-i],i-1)*k.z[i]+1;
```

Η εντολή βρόχου while

Η εντολή βρόχου while είναι μια δομημένη εντολή με παραμέτρους μια έκφραση σε παρενθέσεις και μια εντολή.

Η εκτέλεση της εντολής while ξεκινά με αποτίμηση της έκφρασης. Αν η έκφραση έχει τιμή που αντιστοιχεί στη λογική τιμή «Αληθής», εκτελείται η εντολή που ακολουθεί την έκφραση. Στη συνέχεια η διαδικασία επαναλαμβάνεται με νέα αποτίμηση της έκφρασης, και η εντολή ξαναεκτελείται όσο η τιμή της έκφρασης που υπολογίζεται είναι «Αληθής».
Μόλις η αποτίμηση της έκφρασης δώσει τιμή που αντιστοιχεί στη λογική τιμή «Ψευδής», η εκτέλεση του βρόχου τερματίζεται.

Ένα παράδειγμα εντολής βρόχου δίνεται παρακάτω:

```
while (i > 0) {
      a[i*m] = y % 2;
      k = f(n*m,a,x);
      if (k == 0) break;
      while (j < 10) b[j][i+k] = g(j,m-k)*1.0;
      i = i-1;
}
```

όπου φαίνεται κι ένα παράδειγμα εντολής break.

Η εμφάνιση κάποιας εντολής continue στη σύνθετη εντολή μιας εντολής while οδηγεί τη ροή του κώδικα στην επόμενη επανάληψη, ξεκινώντας με την αποτίμηση της έκφρασης.

Η εντολή βρόχου for

Η εντολή βρόχου for είναι μια δομημένη εντολή της C++600 με παραμέτρους ένα πεδίο επανάληψης και μια εντολή.

Το πεδίο επανάληψης περιέχει σε παρενθέσεις τρεις προαιρετικές εκφράσεις που διαχωρίζονται μεταξύ τους με το χαρακτήρα ‘;’. Άσχετα αν κάποια έκφραση απουσιάζει, η δομή της εντολής δεν αλλάζει.

Η εκτέλεση της εντολής for ξεκινά με αποτίμηση της πρώτης από τις τρεις εκφράσεις, αν αυτή είναι παρούσα. Στη συνέχεια αποτιμάται η δεύτερη έκφραση, αν είναι παρούσα, και αν αυτή δώσει τιμή που αντιστοιχεί στη λογική τιμή «Αληθής», εκτελείται η εντολή που ακολουθεί τις εκφράσεις. Τέλος, αποτιμάται η τρίτη έκφραση, αν είναι παρούσα, και η διαδικασία επαναλαμβάνεται με την αποτίμηση της δεύτερης έκφρασης.
Ο βρόχος τερματίζεται μόλις η δεύτερη έκφραση δώσει τιμή «Ψευδής».
Ας σημειωθεί ότι η πρώτη έκφραση του πεδίου επανάληψης αποτιμάται *μόνο* μία φορά, στην αρχή της εκτέλεσης της εντολής for, ενώ οι υπόλοιπες αποτιμώνται σε κάθε επανάληψη.

Ένα παράδειγμα εντολής βρόχου for δίνεται παρακάτω:

```
for (i = 0; i < n; i++) {
      a[i*m] = y/2;
      if ((k = s*f(n*m,a,x.i)) == 0) continue;
      for (; k > 0; k--) b[j][i+k] = (m-k)*1.0;
}
```

όπου δίνεται κι ένα παράδειγμα χρήσης εντολής ανάθεσης μέσα σε μια έκφραση, όπως κι ένα παράδειγμα εντολής continue.

Η εμφάνιση μιας εντολής continue στη σύνθετη εντολή της εντολής for οδηγεί τη ροή του κώδικα στην επόμενη επανάληψη, ξεκινώντας με την αποτίμηση της *τρίτης* έκφρασης του πεδίου επανάληψης.

Η εντολή επιλογής

Η εντολή επιλογής switch είναι μια δομημένη εντολή με παραμέτρους μια έκφραση σε παρενθέσεις και μία ακολουθία από εντολές, πιθανά προηγούμενες από διαφορετικές ενδεχόμενες τιμές της έκφρασης ή τη λέξη-κλειδί “default”.

Η εκτέλεση της εντολής επιλογής ξεκινά με την αποτίμηση της έκφρασης. Στη συνέχεια ο κώδικας εκτελεί άλμα σε εκείνη την εντολή από την ακολουθία, της οποίας η αναγραφόμενη προηγούμενη τιμή ισούται με την τιμή της έκφρασης. Αν δεν υπάρχει τέτοια εντολή, και υπάρχει εντολή που να ακολουθεί τη λέξη-κλειδί “default”, τότε εκτελείται άλμα σε εκείνη η εντολή. Διαφορετικά δεν εκτελείται καμιά εντολή.

Ένα παράδειγμα εντολής επιλογής δίνεται παρακάτω:

```
switch (x.info%5) {
      case 0: case 1: a[i-2][j].t = 0; break;
      case 4: a[i-2][j+1].t = 1; a[i-2][j-1].u = -1; break;
      default: try_other(x,a);
}
```

Για τις εντολές επιλογής ισχύουν οι ακόλουθοι σημασιολογικοί περιορισμοί:

1. Σε μια εντολή επιλογής, η έκφραση επιλογής πρέπει να είναι βασικού τύπου ή τύπου απαρίθμησης, ενώ οι αναγραφόμενες τιμές πρέπει να έχουν τον ίδιο τύπο με τον τύπο της έκφρασης.
2. Σε οποιαδήποτε περίπτωση εκτελεστεί άλμα προς κάποια εντολή της ακολουθίας, θα εκτελεστούν στη σειρά και οι επόμενες εντολές της ακολουθίας, μέχρι να συναντηθεί εντολή break, οπότε και θα τερματιστεί η εκτέλεση της εντολής switch. Η εντολή break θα πρέπει να αναφέρεται στη συγκεκριμένη εντολή επιλογής, δηλαδή να μην αναφέρεται σε κάποια φωλιασμένη εντολή βρόχου ή εντολή επιλογής, διαφορετικά θα επηρεάσει τη ροή του κώδικα σε σχέση με εκείνη την εντολή. Μέχρι να συναντηθεί κατάλληλη εντολή

break είναι πιθανό η ροή να περάσει και άλλες τιμές επιλογής, χωρίς αυτό να επηρεάζει την εκτέλεση του κώδικα.

3. Πολλαπλές τιμές επιλογής επιτρέπονται με διαδοχικές χρήσεις της λέξης-κλειδί “case”.
4. Στην αρχή της σύνθετης εντολής μιας εντολής switch είναι δυνατό να υπάρχουν δηλώσεις, που όπως σε όλες τις σύνθετες εντολές της C++600 έχουν ορατότητα την εμβέλεια της εντολής.
5. Η σύνταξη της εντολής επιλογής επιτρέπει και μια απλουστευμένη εντολή επιλογής, χωρίς σύνθετη εντολή, αλλά με πιθανά πολλαπλές τιμές επιλογής.