ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

ΑΝΟΙΚΤΑ ακαδημαϊκά μαθήματα ΠΠ

Οντοκεντρικός Προγραμματισμός

Ενότητα 5: Η ΓΛΩΣΣΑ C++

Πίνακες & Δείκτες

ΔΙΔΑΣΚΟΝΤΕΣ: Ιωάννης Χατζηλυγερούδης, Χρήστος Μακρής

Πολυτεχνική Σχολή

Τμήμα Μηχανικών Η/Υ & Πληροφορικής

Πίνακες

Πίνακες

- Τα στοιχεία ενός πίνακα συμπεριφέρονται ως μεταβλητές
  - Ανάθεση/Εκτύπωση

```
c[ 0 ] =  3;
cout << c[ 0 ];
```

- Ο δείκτης ενός πίνακα μπορεί να προκύψει μετά από κάποιο υπολογισμό

`c[ 5 - 2 ]` ίδιο με `c[3]`

Παράδειγμα

Αρχικοποίηση, εκτύπωση πίνακα

```
int main(){
   int n[ 10 ] = { 3, 6, 9, 12, 15 };
   for ( int i = 0; i < 5; i++ )
      cout << n[ i ] << endl;
   return 0;
}
```

Παράδειγμα

Αρχικοποίηση, εκτύπωση πίνακα

```
int main(){
  const int size=5;
  int n[size];
  for ( int i = 0; i < size; i++ )
    n[i] = i*5;
  for ( int i = 0; i < size; i++ )
    cout << n[i] <<" ";
  return 0;
}
```

Μόνο const επιτρέπεται (δεν αλλάζει) για το μέγεθος

```
0 5 10 15 20
```

Πίνακες Χαρακτήρων (Strings)

- Πίνακες χαρακτήρων
- Όλα τα strings τερματίζουν με το null χαρακτήρα (`'\0'`)
- Παραδείγματα
  - **`char string1[] = "hello";`**
    - Το **`Null`** θα προστεθεί στο τέλος
    - **Η μεταβλητή** `string1` έχει 6 στοιχεία
  - **`char string1[] = { 'h', 'e', 'l', 'l', 'o', '\0' };`**
- Η προσπέλαση των στοιχείων γίνεται με τον ίδιο τρόπο

  `String1[ 0 ]` is **`'h'`**

  `string1[ 2 ]` is **`'l'`**

Πέρασμα πίνακα σε συνάρτηση

- Περνάμε το όνομα του πίνακα και προαιρετικά το μέγεθος του πίνακα
  - Παράδειγμα

```
int myArray[ 24 ];

myFunction( myArray, 24 );
```

  - Το μέγεθος του πίνακα είναι καλή πρακτική να περνά ως όρισμα γιατί μας επιτρέπει να έχουμε loops προσπέλασης όλων των στοιχείων εντός της συνάρτησης
- Το πέρασμα του πίνακα γίνεται με αναφορά (by-reference)
  - Η συνάρτηση μπορεί να αλλάξει τις τιμές των στοιχείων του αρχικού πίνακα
  - Το όνομα του πίνακα αντιστοιχεί στη διεύθυνση του πρώτου στοιχείου του πίνακα

Πρωτότυπο συνάρτησης

```
void modifyArray( int b[], int arraySize );
void modifyArray( int [], int );
```

Πολυδιάστατοι πίνακες

- Δύο διαστάσεις
  - **a[i][j]**
  - Πίνακες με γραμμές και στήλες
  - Πρώτα αναφέρουμε τη γραμμή και μετά τη στήλη
  - “Array of arrays”
    - **a[0]** είναι ένας πίνακας 4 στοιχείων
    - **a[0][0]** είναι το πρώτο στοιχείο αυτού του πίνακα
  - Αρχικοποίηση
    - Default τιμή 0
    - Η λίστα αρχικοποιεί κατά γραμμές

```
int a[ 2 ][ 2 ] = { { 1, 2 }, { 3, 4 } };
```

Πολυδιάστατοι πίνακες

```
void printArray( int [][ 3 ] );

int main(){
   int pinakas1[ 2 ][ 3 ] = { { 1, 2, 3 }, { 4, 5, 6 } };
   int pinakas2[ 2 ][ 3 ] = { 1, 2, 3, 4, 5 };
   int pinakas3[ 2 ][ 3 ] = { { 1, 2 }, { 4 } };
   cout << "Pinakas1:" << endl;
   printArray( pinakas1 );
   cout << "Pinakas2:" << endl;
   printArray( pinakas2 ); cout << "Pinakas3:" << endl;
   printArray( pinakas3 );
   return 0;
}

void printArray( int a[][ 3 ] ) {
  for ( int i = 0; i < 2; i++ ) {
    for ( int j = 0; j < 3; j++ )
      cout << a[ i ][ j ] << '.';
    cout << endl;
  }
}
```

```
Pinakas1:
1.2.3.
4.5.6.
Pinakas2:
1.2.3.
4.5.0.
Pinakas3:
1.2.0.
4.0.0.
```

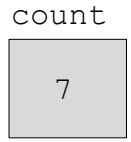
Δείκτες

Δείκτες

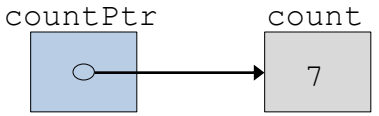
- Δείκτες (Pointers)
  - Ισχυρό χαρακτηριστικό, αλλά δύσκολη η διαχείρισή τους
  - Υλοποιούν ένα είδος pass-by-reference
  - Στενή σχέση με τους πίνακες και τα strings

Δήλωση και αρχικοποίηση μεταβλητών τύπου δείκτη

- Μεταβλητές τύπου δείκτη
  - Αποθηκεύουν διευθύνσεις μνήμης ως τιμές
  - Μια κανονική μεταβλητή αποθηκεύει συγκεκριμένες τιμές (άμεση αναφορά - direct reference)
  - Ένας δείκτης αποθηκεύει τη διεύθυνση μιας μεταβλητής που περιέχει μια συγκεκριμένη τιμή (έμμεση αναφορά - indirect reference)



- Έμμεση αναφορά (Indirection)
  - Η αναφορά μιας τιμής μέσω ενός δείκτη



- Δήλωση δείκτη
  - Το ***** στη δήλωση υποδηλώνει ότι η μεταβλητή είναι δείκτης

```
int *myPtr;
```

- Οι δείκτες μπορούν να χρησιμοποιηθούν με οποιοδήποτε τύπο δεδομένων.
- Αρχικοποίηση δείκτη
  - Αρχικοποιείται στο **0** ή **NULL**

Τελεστές Δεικτών

- **&** (τελεστής διεύθυνσης)
  - Επιστρέφει **τη διεύθυνση** μνήμης του τελεστέου
  - Παράδειγμα

```
int y = 5;
int *yPtr;
yPtr = &y;
```

Τελεστές Δεικτών

- **`*`** (τελεστής έμμεσης αναφοράς ή αποαναφοράς)
  - Επιστρέφει **το αντικείμενο** που δείχνει ο δείκτης
  - **`*yPtr`** επιστρέφει το **`y`** (αφού **`yPtr`** δείχνει στο **`y`**).
  - Ένας αποαναφοροποιημένος δείκτης είναι ένα lvalue

```
*yptr = 9;      // assigns 9 to y
```

- **`*`** και **`&`** είναι αντίστροφοι τελεστές (αλληλοαναιρούνται)

Κλήση συναρτήσεων και πέρασμα ορισμάτων με αναφορά

- Pass-by-reference με ορίσματα δείκτες
  - Περνάμε τη διεύθυνση του ορίσματος χρησιμοποιώντας τον τελεστή διεύθυνσης **&**
  - Οι πίνακες δεν χρειάζονται τον τελεστή **&** αφού το όνομα του πίνακα αντιστοιχεί ήδη σε δείκτη
  - Χρησιμοποιούμε τον τελεστή έμμεσης αναφοράς ***** για την προσπέλαση των τιμών των μεταβλητών εντός της συνάρτησης

Παράδειγμα

```
void fooByValue( int x ) {
  x=x+5;
}
void fooByReference1( int* x ) {
  *x=*x+5;          // χρήση * για να πάρουμε την τιμή
}
void fooByReference2( int& x ) {
  x=x+5;
}
int main(){
  int val = 5;
  fooByValue(val );
  cout << val << endl;     // 5 δεν άλλαξε το val
  fooByReference1(&val);   //   χρήση & για να πάρουμε την διεύθυνση
  cout << val << endl;     // 10 - άλλαξε
  fooByReference2(val);
  cout << val << endl;     // 15 - άλλαξε
  return 0;
}
```

```
5
10
15
```

Χρήση του const

- **`const`** qualifier
  - Δεν επιτρέπει την αλλαγή της τιμής της μεταβλητής
  - Βάζουμε το **`const`** όταν η συνάρτηση δεν χρειάζεται να αλλάξει την τιμή της μεταβλητής
- Αρχή του ελαχίστου δικαιώματος
  - Μια συνάρτηση δεν χρειάζεται να αποκτά παραπάνω δικαιώματα από αυτά που χρειάζεται για την επίτευξη της εργασίας της
- **`const pointers`**
  - Πάντα δείχνουν στην ίδια θέση μνήμης
  - Η περίπτωση του ονόματος ενός πίνακα
  - Πρέπει να αρχικοποιείται όταν δηλώνεται

Πρόσθετο Υλικό

- Μελετήστε και τα παραδείγματα από τα **Κεφάλαια 7, 8** του βιβλίου:
  «C++ How to Program, 9/e Paul & Harvey Deitel»
  http://media.pearsoncmg.com/ph/esm/deitel/cpp_htp_9/code_examples/Code_Examples.zip

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στο πλαίσιο του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αθηνών**» έχει χρηματοδοτήσει μόνο την αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Σημείωμα Ιστορικού Εκδόσεων Έργου

Το παρόν έργο αποτελεί την έκδοση 1.0.1

Σημείωμα Αναφοράς

Σημείωμα Αδειοδότησης

Διατήρηση Σημειωμάτων

Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:

- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων
- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)

μαζί με τους συνοδευόμενους υπερσυνδέσμους.

Σημείωμα Χρήσης Έργων Τρίτων

- Οι διαφάνειες βασίζονται στο βιβλίο «C++ How to Program, 8th Edition, Harvey M. Deitel, Paul J. Deitel, Prentice Hall.»