Επαναληπτική διάλεξη

Δομημένος Προγραμματισμός

Ε. Κατσίρη

2023-24

Δομή ενός απλού προγράμματος

```
#include <stdio.h>                  /* Pre-processor directive */

int main()                          /* main function, start of program */
{
  int year = 2006;                  // a integer variable declaration
  char entry;                       // a character variable

  printf("Welcome to C Programming, Summer %d\n", year);
  printf("Enter 'x' to quit: ");
  scanf("%c", &entry);              // read input from user

  printf("Goodbye!\n");

  return 0;                         // return value to operating system
}
```

Μεταγλώτιση και εκτέλεση

- Ξεκινώντας με τον πηγαίο κώδικα (πρόγραμμα προηγούμενο)
- Προ-επεξεργασία
  - #include <stdio.h>
- Μεταγλώττιση
  - Αντικειμενικός κώδικας
- Σύνδεση
  - Προσθήκη προ-μεταγλωτισμένου κώδικα βιβλιοθήκης
- Εκτέλεση

Μεταβλητές

- Αποθηκεύουν δεδομένα
- Κάθε μεταβλητή περιλαμβάνει:
  - Όνομα (αναγνωριστικό χρήστη)
  - Τύπο (προσδιορίζεται από τη δήλωση της συνάρτησης)
  - Μέγεθος (προσδιορίζεται από τον τύπο)
  - Τιμή (τα δεδομένα που αποθηκεύονται στη μεταβλητή)
  - Διεύθυνση
- Πίνακας συμβόλων
  - Μεταβλητή -> <όνομα, τύπος, μέγεθος κλπ>
  - Παράγεται από τη μεταγλωτιση

Αναγνωριστικά

- Πρόκειται για τα ονόματα μεταβλητών, συναρτήσεων κλπ
- Τα αναγνωριστικά που καθορίζονται από τον χρήστη μπορούν να αποτελούνται από γράμματα, ψηφία και underscore.
- Πρέπει να αρχίζουν με μη-ψηφίο
- Τα αναγνωριστικά είναι ευαίσθητα στα κεφαλαία και τα πεζά
  - (count και Count είναι διαφορετικές μεταβλητές)
- Οι δεσμευμένες λέξεις (λέξεις-κλειδιά) δεν μπορούν να χρησιμοποιηθούν ως αναγνωριστικά.

Δήλωση μεταβλητών

- Οι μεταβλητές πρέπει να δηλώνονται πριν χρησιμοποιηθούν
- format: typeName variableName1, variableName2, ...;

```
int numStudents;        // variable of type integer
double weight;          // variable of type double
char letter;            // variable of type character

// Examples of multiple variables of the same type in single
//  declaration statements

int test1, test2, finalExam;
double average, gpa;
```

Ατομικοί τύποι δεδομένων

- Ακέραιοι τύποι
  - Char (1 byte)
  - Short
  - Int (4 byte)
  - Long
  - Κανονικοί και unslgned τύποι
- Τύποι κινητής υποδιαστολής
  - Float
  - Double
  - Long double
- Μεγέθη
  - Συναρτηση sizeof()
  - Εξαρτώνται από την υλοποίηση

Αρχικοποίηση μεταβλητών

- Το να δηλώσουμε μια μεταβλητή σημαίνει να πούμε στον μεταγλωττιστή ότι υπάρχει και να κρατήσουμε μνήμη γι' αυτήν.
- Η αρχικοποίηση μιας μεταβλητής είναι η φόρτωση μιας τιμής σε αυτήν για πρώτη φορά.
- Εάν μια μεταβλητή δεν έχει αρχικοποιηθεί, περιέχει ό,τι bit υπάρχει ήδη στη μνήμη στη θέση της μεταβλητής (δηλαδή σκουπίδια)
- Γίνεται μέσω εντολών ανάθεσης
- Ακόμη και στην ίδια γραμμή

```
int numStudents;
double weight;
char letter;

numStudents = 10;
weight = 160.35;
letter = 'A';
```

```
int numStudents = 10;
double weight = 160.35;
char letter = 'A';

int test1 = 96, test2 = 83, finalExam = 91;
double x = 1.2, y = 2.4, z = 12.9;
```

Σταθερές μεταβλητές

- Δεν μπορούν να αλλάξουν τιμή μετά την αρχικοποίηση
- Αρχικοποιούνται και δηλώνονται στην ίδια γραμμή
- Κεφαλαία

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

// this one is illegal, because it's not
//  initialized on the same line
const int LIMIT;                          // BAD!!!
LIMIT = 20;
```

Αλλα στοιχεία

- Συμβολικές σταθερές

```
#define PI 3.14159
#define DOLLAR '$'
#define MAXSTUDENTS 100
```

- Literals
  - Σταθερές μεταβλητές
  - Οι τιμές των μεταβλητών
  - integer literal -- (4, -10, 18)
  - floating point literal -- (4.5, -12.9, 5.0)
  - character literal -- ('F', 'a', '\n')
  - string literal -- ("Hello", "Bye", "Wow!\n")

Printf()

- printf (*format_string*, *list_of_expressions*);

```
int x = 50, y = 8, z = 5;                    // three integer variables

printf("There are %d states in the U.S.\n", x );
printf("The final score was %d to %d\n", y, z );

// the output from this code is:
//   There are 50 states in the U.S.
//   The final score was 8 to 5
```

scanf()

- scanf(*format_string*, *list_of_variable_addresses*);
- Διαβάζει δεδομένα από την τυπική είσοδο

```
int numStudents, month, day;
printf("Please enter the number of students in the class: ");
scanf("%d", &numStudents);

printf("Please enter your birth month, followed by the day: ");
scanf("%d %d", &month, &day);

// at this point, the user has typed in three values, stored now in
//  numStudents, month, and day -- respectively
```

Τελεστές

```
x == y        // x is equal to y
x != y        // x is not equal to y
x < y         // x is less than y
x <= y        // x is less than or equal to y
x > y         // x is greater than y
x >= y        // x is greater than or equal to y
```

```
int x, y = 5, z;

z = 10;                 // assignment operator (binary)
x = y + z;              // addition (binary operator)
x = -y;                 // -y is a unary operation (negation)
x++;                    // unary (increment)

x = a + b + c - d + e;          // cascading arithmetic operators
x = y = z = 3;                  // cascading assignment operators

x = a + b * c;          // b * c happens first, since * has higher
                        //  precedence than +
```

Αυξητικοί και μειωτικοί

```
++x;  // pre-increment (value of expression is new value of x)
x++;  // post-increment (value of expression is OLD value of x)
            // shortcuts for x = x + 1

--x;  // pre-decrement (returns new value)
x--;  // post-decrement (returns OLD value)
            // shortcuts for x = x - 1
```

Δομές ελέγχου / επανάληψης

- `if`
- `if/else`
- `switch`

- `while`
- `do/while`
- `for`

Συναρτήσεις

- Η συνάρτηση είναι ένα επαναχρησιμοποιήσιμο τμήμα ενός προγράμματος, που μερικές φορές ονομάζεται διαδικασία ή υπορουτίνα.
- Σαν ένα μίνι-πρόγραμμα (ή υποπρόγραμμα) από μόνη της.
- Μπορεί να δέχεται ειδικές εισόδους (ορίσματα)
- Μπορεί να παράγει μια τιμή απάντησης (τιμή επιστροφής)
- Παρόμοια με την ιδέα της συνάρτησης στα μαθηματικά

Συναρτήσεις (2)

- Δήλωση συνάρτησης

```
int Sum(int x, int y, int z);
double Average (double a, double b, double c);
int InOrder(int x, int y, int z);
int DoTask(double a, char letter, int num);
```

- Ορισμός συνάρτησης

```
double Average (double a, double b, double c)
// add the parameters, divide by 3, and return the result
{
   return (a + b + c) / 3.0;
}
```

- Εμβέλεια αναγνωριστικών
- Void συναρτήσεις και άδεια ορίσματα

```
char GetALetter();                          // no parameters
void PrintQuotient(int x, int y);           // void return type
void KillSomeTime();                        // both
```

Απαριθμήσιμοι τύποι

```
enum Days {SUN, MON, TUE, WED, THUR, FRI, SAT};

enum Days today, tomorrow, yesterday;   // create three variables,
                                        //   of type 'Days'
today = MON;
if (today == SUN)
    yesterday = SAT;
if (tomorrow == FRI)
    printf("Today is Thursday!");
```

Πίνακες

- Ένας πίνακας είναι μια ευρετηριασμένη συλλογή στοιχείων δεδομένων του ίδιου τύπου.
- 1) Ευρετηριασμένη σημαίνει ότι τα στοιχεία του πίνακα είναι αριθμημένα (ξεκινώντας από το 0).
- 2) Ο περιορισμός του ίδιου τύπου είναι σημαντικός, επειδή οι πίνακες αποθηκεύονται σε διαδοχικά κελιά μνήμης. Κάθε κελί πρέπει να έχει τον ίδιο τύπο (και, επομένως, το ίδιο μέγεθος).

Πίνακες

- Δήλωση πινάκων

```
int list[30];            // an array of 30 integers
char name[20];           // an array of 20 characters
double nums[50];         // an array of 50 decimals
int table[5][10];        // a two dimensional array of integers
```

- Αρχικοποίηση πινάκων

```
int list[4] = {2, 4, 6, 8};
char letters[5] = {'a', 'e', 'i', 'o', 'u'};
double numbers[3] = {3.45, 2.39, 9.1};
int table[3][2] = {{2, 5} , {3,1} , {4,9}};
```

Αλφαριθμητικά

- Αλφαριθμητικά (strings)

```
char name[7] = "Johnny";
```

- Χρήση πινάκων

```
list[3] = 6;                    // assign value 6 to array item with index 3
printf("%.2f", nums[2]);        // output array item with index 2
list[x] = list[x+1];
```

- Αντιγραφή πινάκων

```
int i;
for (i = 0; i < 5; i++)
  list1[i] = list2[i];
```

Πίνακες και συναρτήσεις

```
void PrintArray (int arr[], int size)
{
  int i;
  for (i = 0; i < size; i++)
    printf("%d ", arr[i]);
}
```

```
int list[5] = {2, 4, 6, 8, 10};

PrintArray(list, 5);          // will print: 2 4 6 8 10
```

- Η μεταβλητή arr λειτουργεί ως το όνομα του τοπικού πίνακα μέσα στη συνάρτηση
- Δεν υπάρχει αριθμός μέσα στις αγκύλες.
- int [] υποδηλώνει ότι πρόκειται για παράμετρο πίνακα, για έναν πίνακα τύπου int
- Συνήθως είναι καλή ιδέα να περάσετε και το μέγεθος του πίνακα, ως άλλη παράμετρο.
- Αυτό βοηθάει στο να λειτουργήσει μια συνάρτηση για πίνακα οποιουδήποτε μεγέθους

typedef

```
typedef enum bool
{false, true} bool;              // bool is now a type name, and it has
                                 //  false and true as its values.
```

```
typedef int List[10];            // List is now a type that means
                                 //  "10 element array of type int"

List x;                          // x is an integer array, size 10
List y;                          // y is an integer array, size 10
```

Δείκτες

- Ένας δείκτης είναι μια μεταβλητή που αποθηκεύει μια διεύθυνση μνήμης.
- Οι δείκτες χρησιμοποιούνται για την αποθήκευση των διευθύνσεων άλλων μεταβλητών ή στοιχείων μνήμης.
- Οι δείκτες είναι πολύ χρήσιμοι για έναν άλλο τύπο μεταβίβασης παραμέτρων, ο οποίος συνήθως αναφέρεται ως Pass By Address.
- Οι δείκτες είναι απαραίτητοι για τη δυναμική κατανομή μνήμης.

Δείκτες

- Δήλωση δεικτών

```
double * dptr;          // a pointer to a double
char * c1;              // a pointer to a character
float * fptr;           // a pointer to a float

int * p, * q, * r;              // three pointers-to-int
int * p, q, r;                  // p is a pointer, q and r are ints
```

- Ας υποθέσουμε ότι ο ptr είναι ακέραιος δείκτης.
- Ας υποθέσουμε ότι αποθηκεύει τη διεύθυνση 1234. Υποθέστε ότι ο ακέραιος που είναι αποθηκευμένος στη διεύθυνση 1234 έχει την τιμή 99.

```
printf("The pointer is: %p\n", ptr);  // prints the pointer
printf("The target is: %d\n", *ptr);  // prints the target


// Output:
//  The pointer is: 1234
//  The target is: 99
```

Αναφορά δείκτη!

Δείκτες

- Αρχικοποίηση δεικτών

```
int * p = 0;     // okay.  assignment of null pointer to p
int * q;
q = 0;           // okay.  null pointer again.
int * z;
z = 900;         // BAD!  cannot assign other literals to pointers!
double * dp;
dp = 1000;       // BAD!


if (ptr != 0)              // safe to dereference
   printf("%d", *ptr);


int * ptr1, * ptr2;          // two pointers of type int

ptr1 = ptr2;                 // can assign one to the other
                             //  now they both point to the same place
```

- Οι κανονικές (τυπικές) παράμετροι μιας συνάρτησης είναι **pass-by-value**
- Μια τυπική παράμετρος μιας συνάρτησης είναι μια τοπική μεταβλητή που θα περιέχει ένα αντίγραφο της τιμής του ορίσματος που έχει περάσει.
- Οι αλλαγές που γίνονται στην τοπική μεταβλητή της παραμέτρου δεν επηρεάζουν το αρχικό όρισμα
- Εάν ένας τύπος δείκτη χρησιμοποιείται ως τύπος παραμέτρου συνάρτησης, τότε αντί αυτού αποστέλλεται στη συνάρτηση μια πραγματική διεύθυνση.
- Μια τέτοια παράμετρος θα περιέχει ένα αντίγραφο της διεύθυνσης που αποστέλλεται από τον καλούντα, αλλά όχι ένα αντίγραφο των δεδομένων.
- Όταν οι διευθύνσεις (δείκτες) διαβιβάζονται σε συναρτήσεις, η συνάρτηση θα μπορούσε να επηρεάσει πραγματικές μεταβλητές που υπάρχουν στην εμβέλεια του καλούντος.
- Αυτό λέγεται **pass-by-address**

Δείκτες

- Pass by Address

```
void SquareByAddress(int * ptr)
// Note that this function doesn't return anything.  void function.
{
   *ptr = (*ptr) * (*ptr);              // modifying the target, *ptr
}

int main()
{
  int num = 4;
  printf("num = %d\n", num);            // num = 4
  SquareByAddress(&num);                // address of num passed in
  printf("num = %d\n", num);            // num = 16
}
```

Δείκτες και Πίνακες

Με μια κανονική δήλωση πίνακα, λαμβάνετε δωρεάν έναν δείκτη.
Το όνομα του πίνακα λειτουργεί ως δείκτης στο πρώτο στοιχείο του πίνακα.

```
int list[10];   // the variable list is a pointer
                // to the first integer in the array
int * p;        // p is a pointer.  It has the same type as list.
p = list;       // legal assignment.  Both pointers to ints.
```

Τώρα, θα μπορούσαμε να χρησιμοποιήσουμε το p ως όνομα του πίνακα!

```
list[3] = 10;
p[4] = 5;
printf("%d", list[6]);
printf("%d", p[6]);
```

Αλφαριθμητικά

- Μια συμβολοσειρά τύπου C είναι ένας πίνακας χαρακτήρων που τελειώνει με τον χαρακτήρα null
- Ο χαρακτήρας null είναι σιωπηρά μέρος κάθε αλφαριθμητικού.
- Όλα τα αλφαριθμητικά είναι σε διπλά εισαγωγικά
- Οι συμβολοσειρές τύπου C σχετίζονται στενά με δείκτες τύπου char (δηλ. τύπου (char *)), καθώς το όνομα ενός πίνακα χαρακτήρων είναι ένας δείκτης στο πρώτο στοιχείο.

Συναρτήσεις αλφαριθμητικών

- Η **ctype.h** είναι μια βιβλιοθήκη που περιέχει χρήσιμες συναρτήσεις χειρισμού χαρακτήρων

- **Conversion functions**: These return the ascii value of a character
  - `int toupper(int c)` - returns the uppercase version of `c` if it's a lowercase letter, otherwise returns `c` as is
  - `int tolower(int c)` - returns the lowercase version of `c` if it's an uppercase letter, otherwise returns `c` as is
- **Query Functions**: These all return true (non-zero) or false (0), in answer to the question posed by the function's name.
  - `int isdigit(int c)` - decides whether the parameter is a digit (0-9)
  - `int isalpha(int c)` - decides whether the character is a letter (a-z, A-Z)
  - `int isalnum(int c)` - digit or a letter?
  - `int islower(int c)` - lowercase digit? (a-z)
  - `int isupper(int c)` - uppercase digit? (A-Z)
  - `int isxdigit(int c)` - hex digit character? (0-9, a-f)
  - `int isspace(int c)` - white space character?
  - `int iscntrl(int c)` - control character?
  - `int ispunct(int c)` - printing character other than space, letter, digit?
  - `int isprint(int c)` - printing character (including ' ')?
  - `int isgraph(int c)` - printing character other than ' ' (space)?

Δηλώσεις αλφαριθμητικών

```
char name[20] = "Marvin Dipwart";

char* greeting = "Hello";

const char* greeting = "Hello";

name[1] = 'e';          // name is now "Mervin Dipwart"

greeting[1] = 'u';      // ILLEGAL!

char greeting[20] = "Hello, World";
printf("%s", greeting);                 // prints "Hello, World"

char lastname[20];
scanf("%s", lastname);          // reads a string into the array 'lastname'     // if the user types "van Buren", the
                                //  adds the null character automatically       //  lastname array now stores "van"
```

Συναρτήσεις I/O

- **`int getchar()`** -- reads the next character from standard input and returns it as an integer (ascii value)
- **`char* gets(char* s)`** -- reads characters into the array `s` until newline or end-of-file character. Appends null-character automatically
  - This is a function that would allow input of a whole sentence, not just one word. Stops at newline, not at all white space.
- **`int putchar(int c)`** -- prints one character (`c`) to standard output
- **`int puts(const char* s)`** -- prints the string `s`, followed by a newline
- These two functions are just like `printf` and `scanf`, except that they both have one extra parameter at the *beginning* -- a string. Instead `s` (or read from `s`):
  - **`sprintf(char* s, const char* format, ... )`**
  - **`sscanf(char* s, const char* format, ... )`**

Βιβλιοθήκη string.h

- **strcpy** - copies the contents of one string to another
- **strcat** - concatenates one string with another
- **strcmp** - compares two strings and determines their lexicographical order
- **strncpy**, **strncat**, **strncmp** - these do the same as the three listed above, but only up to the first n characters of the string
- **strlen** - calculates the length of a string
- **strtok** -- string tokenization. Used to break up a string into smaller ones

```
char* strcpy( char * s1, const char * s2 );
```

Δομές

- Έχουμε πολλούς απλούς τύπους για την αποθήκευση μεμονωμένων στοιχείων όπως αριθμοί, χαρακτήρες. Αλλά είναι αυτό πραγματικά αρκετό για την αποθήκευση πιο σύνθετων πραγμάτων, όπως αρχεία ασθενών, βιβλία διευθύνσεων, πίνακες κ.λ.π;
- Ο **ορισμός μιας δομής** είναι σαν ένα σχέδιο της δομής. Δεν καταλαμβάνει ο ίδιος αποθηκευτικό χώρο -- απλά καθορίζει πώς θα μοιάζουν οι μεταβλητές αυτού του τύπου δομής.
- Μια πραγματική **μεταβλητή δομής** είναι σαν ένα κουτί με πολλαπλά πεδία δεδομένων στο εσωτερικό του. Σκεφτείτε την ιδέα μιας βάσης δεδομένων μαθητών. Μια εγγραφή μαθητή περιέχει πολλαπλές πληροφορίες (όνομα, διεύθυνση, SSN, GPA, κ.λπ.)

Δομές

- Ιδιότητες μιας δομής:
  - Τα εσωτερικά στοιχεία μπορεί να είναι διαφόρων τύπων δεδομένων
  - Η σειρά των στοιχείων είναι αυθαίρετη (δεν υπάρχει ευρετηρίαση, όπως στους πίνακες)
  - Σταθερό μέγεθος, με βάση τα συνδυασμένα μεγέθη των εσωτερικών στοιχείων

Ορισμός δομής

```
/* A structure representing the parts of a fraction (a rational number) */
struct fraction
{
   int num;            // the numerator of the fraction
   int denom;          // the denominator of the fraction
};

/* A structure representing a record in a student database */
struct student
{
   char fName[20];             // first name
   char lName[20];             // last name
   int socSecNumber;           // social security number
   double gpa;                 // grade point average
};
```

Χρήση δομής

```
struct student s1 = {"John", "Smith", 123456789, 3.75};
struct student s2 = {"Alice", "Jones", 123123123, 2.66};
```

```
struct fraction f1, f2;

f1.num = 4;                    // set f1's numerator to 4
f1.denom = 5;                  // set f1's denominator to 5
f2.num = 3;                    // set f2's numerator to 3
f2.denom = 10;                 // set f2's denominator to 10

printf("f1 = %d/%d\n", f1.num, f1.denom);     // prints 4/5
printf("f2 = %d/%d\n", f2.num, f2.denom);     // prints 3/10
```

```
struct student sList[10];     // array of 10 students

// set first student's data: (John Smith, SSN: 123456789, GPA: 3.75)
strcpy(sList[0].fName, "John");
strcpy(sList[0].lName, "Smith");
sList[0].socSecNumber = 123456789;
sList[0].gpa = 3.75;

// assume there's more code here that initializes other students

// This loop prints all 10 students -- their names and their GPA
int i;
for (i = 0; i < 10; i++)
   printf("%s %s: %.2f\n", sList[i].fName, sList[i].lName, sList[i].gpa);
```

Ασκήσεις εξάσκησης

1. Γράψτε μια συνάρτηση με όνομα **Sum** που δέχεται έναν πίνακα τύπου double και επιστρέφει το άθροισμα των στοιχείων του.
2. Γράψτε έναν ορισμό δομής με το όνομα **book**, ο οποίος ενσωματώνει τις ακόλουθες πληροφορίες: τίτλος βιβλίου (μια συμβολοσειρά), συγγραφέας βιβλίου (μια συμβολοσειρά), έτος έκδοσης (ακέραιος αριθμός), είδος (π.χ. μυστήριο, φαντασία, ρομάντζο κ.λπ.) -- (μια συμβολοσειρά), τιμή (double).
3. Γράψτε ένα πρόγραμμα που να διαβάζει συνεχώς τις λέξεις που εισάγονται στο πληκτρολόγιο (σταματώντας μόλις ο χρήστης πληκτρολογήσει τον χαρακτήρα νέας γραμμής) και στη συνέχεια να εκτυπώνει πόσες φορές εμφανίζεται η λέξη ” **τελείως**" στην είσοδο.
4. Γράψτε μια συνάρτηση που ονομάζεται **power** η οποία δέχεται δύο ακέραιους αριθμούς, που αντιπροσωπεύουν μια βάση και έναν εκθέτη, και επιστρέφει την τιμή: $base^{exponent}$