Διάλεξη 23η: Επιπλέον στοιχεία της C

Τμήμα Επιστήμης Υπολογιστών, Πανεπιστήμιο Κρήτης

Εισαγωγή στην Επιστήμη Υπολογιστών

Οργάνωση του κώδικα

- Ένα πρόγραμμα μπορεί να βρίσκεται σε πολλά αρχεία
- Καλύτερη οργάνωση του κώδικα
- Χρήσιμο/απαραίτητο, ειδικά σε μεγάλα προγράμματα
  - Το linux kernel έχει 28 εκατομμύρια γραμμές κώδικα
  - Τα Windows 11 έχουν 50 εκατομμύρια γραμμές κώδικα
  - Το MacOS έχει 80
  - Ο firefox έχει 2.1 εκατομμύρια γραμμές κώδικα σε 11000 αρχεία
- Χωριστή μεταγλώττιση (compile) και σύνδεση (link)
  - Ο κώδικας βρίσκεται σε αρχεία `.c`
  - Οι δηλώσεις και ορισμοί τύπων βρίσκονται σε αρχεία `.h` (επικεφαλίδας–header)
  - Όλα τα αρχεία κώδικα που χρησιμοποιούν τύπους, συναρτήσεις, μεταβλητές, κλπ., δηλωμένα σε ένα header, το εισάγουν με `#include "file.h"`
  - Οι δηλώσεις συναρτήσεων και μεταβλητών γίνονται με `extern`

Παράδειγμα

intlist.h

```
struct list_node {
  int data;
  struct list_node *next;
};
typedef struct list_node node_t;

extern node_t *root;

extern node_t *list_insert_before(node_t *, int);
extern void list_insert_after(node_t *, int);
extern node_t *list_find(node_t *, int);
extern node_t *list_nth(node_t *, int);
extern node_t *list_remove_node(node_t *, node_t *);
extern void list_delete(node_t *);
```

Παράδειγμα (2)

intlist.c

```
#include ”intlist.h”

node_t *root = NULL;

node_t *list_insert_before(node_t *list, int data) { ... }
void list_insert_after(node_t *list, int data) { ... }
node_t *list_find(node_t *list, int data) { ... }
node_t *list_nth(node_t *list, int position) { ... }
node_t *list_remove_node(node_t *list, node_t *node) { ... }
void list_delete(node_t *list) { ... }
```

Παράδειγμα (3)

main.c

```
#include ”intlist.h”

int main() {
  node_t *ptr;

  list_insert_after(root, 10);
  root = list_insert_before(root, 42);
  ptr = list_find(root, 10);
  printf(”found %d!\n”, ptr->data);
  return 0;
}
```

Παράδειγμα (4)

- Μεταγλώττιση κάθε αρχείου χωριστά σε object code

Μεταγλώττιση

```
$ gcc -c intlist.c -o intlist.o
$ gcc -c main.c -o main.o
```

- Σύνδεση όλων των object αρχείων για την παραγωγή του εκτελέσιμου

Σύνδεση

```
$ gcc main.o intlist.o -o program
$ ./program
found 10!
$
```

Bits

- Δεκαδικό σύστημα:

$$1993_{10} = 1 \times 10^3 + 9 \times 10^2 + 9 \times 10^1 + 3 \times 10^0$$

- Δυαδικό σύστημα:

$$1101_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

- Οκταδικό σύστημα:

$$7103_8 = 7 \times 8^3 + 1 \times 8^2 + 0 \times 8^1 + 3 \times 8^0$$

- Δεκαεξαδικό σύστημα:

$$F1A7_{16} = 15 \times 16^3 + 1 \times 16^2 + 10 \times 16^1 + 7 \times 16^0$$

Τελεστές Bits

- Οι τελεστές bit εφαρμόζονται σε κάθε ένα bit του κάθε (ακέραιου) τελεσταίου
- Τελεστής NOT: `~`
  - `~10000111 == 01111000`
- Τελεστής AND: `&`
  - `(11100011 & 10000111) == 10000011`
- Τελεστής OR: `|`
  - `(11100011 | 10000111) == 11100111`
- Τελεστής XOR: `^`
  - `(11100011 ^ 10000111) == 01100100`

Τελεστές Ολίσθησης

- Τελεστής shift left: `<<`
- Τελεστής shift right: `>>`
- Ολισθαίνουν τα bits του πρώτου τελεσταίου δεξιά ή αριστερά κατά τόσες θέσεις όσες καθορίζει ο δεύτερος τελεσταίος
- Τα καινούρια bits είναι πάντα 0
  - Το shift right διατηρεί το πρόσημο σε `signed int`
- Παράδειγμα
  - `(10000000 >> 2) == 00100000`
  - `(00000011 << 3) == 00011000`

Παράδειγμα

countbits.c

```
int count_bits(int x)
{
  int n;
  for(n = 0; x != 0; ++n) {
    x &= (x -1);
  }
  return n;
}
```

Πεδία bit

- Σε δομές που περιέχουν πεδία με περιορισμένες τιμές
- Μπορούμε να ορίσουμε πόσα bits καταλαμβάνει ένα πεδίο

```
struct foo {
  unsigned year     :11;
  unsigned is_valid :1;
  unsigned percent  :7;
  unsigned          :0;
  unsigned field1   :3;
  signed   field2   :10;
};
```

- Τα bit fields δεν έχουν διεύθυνση (δεν εφαρμόζεται ο τελεστής &)
- Η σειρά των bit fields στη μνήμη δεν καθορίζεται από τη C
  - Μπορεί να αλλάξει ανάλογα με τον compiler και τον επεξεργαστή