# Διάλεξη 3η: Τύποι Μεταβλητών, Τελεστές, Είσοδος/Έξοδος

Τμήμα Επιστήμης Υπολογιστών, Πανεπιστήμιο Κρήτης

Εισαγωγή στην Επιστήμη Υπολογιστών

Βασίζεται σε διαφάνειες του Κ. Παναγιωτάκη

# Μεταβλητές

- Οι μεταβλητές είναι "επώνυμες" θέσεις μνήμης
- Στη C όλες οι μεταβλητές πρέπει να δηλώνονται
- Δηλώσεις:
  - Της μορφής: "`τύπος όνομα;`"
  - Π.χ: `int my_first_variable;`
- Ονόματα μεταβλητών
  - Καλύτερα να είναι μεγάλα και επεξηγηματικά παρά μικρά
    - Π.χ: `sum`, `percentage_downloaded`, και όχι `x42`, `giorgos1993`
  - Μέχρι 31 χαρακτήρες, ξεκινούν από γράμμα ή "_", μπορούν να περιέχουν αριθμούς
  - Είναι case-sensitive, π.χ: `vrbl1` $\neq$ `Vrbl1`
  - Δεν πρέπει να είναι δεσμευμένες λέξεις
    - Π.χ: `if`, `else`, `while`, `for`, `break`, `continue`, `int`, `float`, `double`, `char`, ...
- Καλό είναι να αρχικοποιούνται:
  - `int my_first_variable = 42`
- Δήλωση πολλών μεταβλητών
  - `int x, y = 0, z = 42;`

# Εμβέλεια

- Οι μεταβλητές μπορεί να είναι:
  - Τοπικές (local)
    - Έχει πρόσβαση σ'αυτές μόνο η συνάρτηση που τις δηλώνει
    - Δημιουργούνται στην αρχή της συνάρτησης
    - Στο τέλος της συνάρτησης "χάνονται"
  - Καθολικές (global)
    - Έχουν πρόσβαση σ'αυτές όλες οι συναρτήσεις που δηλώνονται στο ίδιο αρχείο
    - Δημιουργούνται στην αρχή της εκτέλεσης
    - Διατηρούνται καθ' όλη την εκτέλεση του προγράμματος
    - Προσοχή στη χρήση τους

# Παράδειγμα: Καθολικές και τοπικές μεταβλητές

## variables.c

```
int total_calls = 0;

int next_of(int x)
{
  x = x + 1;
  total_calls = total_calls + 1;
  return x;
}

int main()
{
  int x = 0;
  x = next_of(0);
  x = next_of(x);
  return 0;
}
```

# Παράδειγμα: Καθολικές και τοπικές μεταβλητές

Δήλωση καθολικής μεταβλητής

variables.c

```
int total_calls = 0;

int next_of(int x)
{
  x = x + 1;
  total_calls = total_calls + 1;
  return x;
}

int main()
{
  int x = 0;
  x = next_of(0);
  x = next_of(x);
  return 0;
}
```

# Παράδειγμα: Καθολικές και τοπικές μεταβλητές

variables.c

```
int total_calls = 0;

int next_of(int x)
{
  x = x + 1;
  total_calls = total_calls + 1;
  return x;
}

int main()
{
  int x = 0;
  x = next_of(0);
  x = next_of(x);
  return 0;
}
```

Συνάρτηση που επιστρέφει τον επόμενο ακέραιο

# Παράδειγμα: Καθολικές και τοπικές μεταβλητές

variables.c

```
int total_calls = 0;

int next_of(int x)
{
  x = x + 1;
  total_calls = total_calls + 1;
  return x;
}

int main()
{
  int x = 0;
  x = next_of(0);
  x = next_of(x);
  return 0;
}
```

Παράμετρος της next_of, λειτουργεί σαν τοπική μεταβλητή

# Παράδειγμα: Καθολικές και τοπικές μεταβλητές

variables.c

```
int total_calls = 0;

int next_of(int x)
{
  x = x + 1;
  total_calls = total_calls + 1;
  return x;
}

int main()
{
  int x = 0;
  x = next_of(0);
  x = next_of(x);
  return 0;
}
```

Αυξάνει τη μεταβλητή x κατά 1

# Παράδειγμα: Καθολικές και τοπικές μεταβλητές

variables.c

```
int total_calls = 0;

int next_of(int x)
{
  x = x + 1;
  total_calls = total_calls + 1;
  return x;
}

int main()
{
  int x = 0;
  x = next_of(0);
  x = next_of(x);
  return 0;
}
```

Αυξάνει τη μεταβλητή total_calls κατά 1

# Παράδειγμα: Καθολικές και τοπικές μεταβλητές

variables.c

```
int total_calls = 0;

int next_of(int x)
{
  x = x + 1;
  total_calls = total_calls + 1;
  return x;
}

int main()
{
  int x = 0;
  x = next_of(0);
  x = next_of(x);
  return 0;
}
```

Επιστρέφει την τιμή του x

# Παράδειγμα: Καθολικές και τοπικές μεταβλητές

variables.c

```
int total_calls = 0;

int next_of(int x)
{
  x = x + 1;
  total_calls = total_calls + 1;
  return x;
}

int main()
{
  int x = 0;
  x = next_of(0);
  x = next_of(x);
  return 0;
}
```

Συνάρτηση main: αρχή της εκτέλεσης

# Παράδειγμα: Καθολικές και τοπικές μεταβλητές

variables.c

```
int total_calls = 0;

int next_of(int x)
{
  x = x + 1;
  total_calls = total_calls + 1;
  return x;
}

int main()
{
  int x = 0;
  x = next_of(0);
  x = next_of(x);
  return 0;
}
```

Τοπική μεταβλητή της main, δήλωση και αρχικοποίηση

# Παράδειγμα: Καθολικές και τοπικές μεταβλητές

variables.c

```
int total_calls = 0;

int next_of(int x)
{
  x = x + 1;
  total_calls = total_calls + 1;
  return x;
}

int main()
{
  int x = 0;
  x = next_of(0);
  x = next_of(x);
  return 0;
}
```

Πρώτη κλήση της next_of με παράμετρο 0

# Παράδειγμα: Καθολικές και τοπικές μεταβλητές

variables.c

```
int total_calls = 0;

int next_of(int x)
{
  x = x + 1;
  total_calls = total_calls + 1;
  return x;
}

int main()
{
  int x = 0;
  x = next_of(0);
  x = next_of(x);
  return 0;
}
```

Δεύτερη κλήση της next_of
με παράμετρο το x της main

# Παράδειγμα: Καθολικές και τοπικές μεταβλητές

variables.c

total_calls = 0

```
int total_calls = 0;

int next_of(int x)
{
  x = x + 1;
  total_calls = total_calls + 1;
  return x;
}

int main()
{
  int x = 0;
  x = next_of(0);
  x = next_of(x);
  return 0;
}
```

# Παράδειγμα: Καθολικές και τοπικές μεταβλητές

## variables.c

```
int total_calls = 0;

int next_of(int x)
{
  x = x + 1;
  total_calls = total_calls + 1;
  return x;
}

int main()
{
  int x = 0;
  x = next_of(0);
  x = next_of(x);
  return 0;
}
```

total_calls = 0

# Παράδειγμα: Καθολικές και τοπικές μεταβλητές

## variables.c

```
int total_calls = 0;

int next_of(int x)
{
  x = x + 1;
  total_calls = total_calls + 1;
  return x;
}

int main()
{
  int x = 0;
  x = next_of(0);
  x = next_of(x);
  return 0;
}
```

total_calls = 0, x = 0

# Παράδειγμα: Καθολικές και τοπικές μεταβλητές

variables.c

```
int total_calls = 0;

int next_of(int x)
{
  x = x + 1;
  total_calls = total_calls + 1;
  return x;
}

int main()
{
  int x = 0;
  x = next_of(0);
  x = next_of(x);
  return 0;
}
```

total_calls = 0,
x (main) = 0,
x (next_of) = 0

# Παράδειγμα: Καθολικές και τοπικές μεταβλητές

variables.c

```
int total_calls = 0;

int next_of(int x)
{
  x = x + 1;
  total_calls = total_calls + 1;
  return x;
}

int main()
{
  int x = 0;
  x = next_of(0);
  x = next_of(x);
  return 0;
}
```

total_calls = 0,
x (main) = 0,
x (next_of) = 0

# Παράδειγμα: Καθολικές και τοπικές μεταβλητές

variables.c

```
int total_calls = 0;

int next_of(int x)
{
  x = x + 1;
  total_calls = total_calls + 1;
  return x;
}

int main()
{
  int x = 0;
  x = next_of(0);
  x = next_of(x);
  return 0;
}
```

total_calls = 0,
x (main) = 0,
x (next_of) = 1

# Παράδειγμα: Καθολικές και τοπικές μεταβλητές

variables.c

```
int total_calls = 0;

int next_of(int x)
{
  x = x + 1;
  total_calls = total_calls + 1;
  return x;
}

int main()
{
  int x = 0;
  x = next_of(0);
  x = next_of(x);
  return 0;
}
```

total_calls = 1,
x (main) = 0,
x (next_of) = 1

# Παράδειγμα: Καθολικές και τοπικές μεταβλητές

variables.c

```
int total_calls = 0;

int next_of(int x)
{
  x = x + 1;
  total_calls = total_calls + 1;
  return x;
}

int main()
{
  int x = 0;
  x = next_of(0);
  x = next_of(x);
  return 0;
}
```

total_calls = 1,
x (main) = 0,
x (next_of) = 1

# Παράδειγμα: Καθολικές και τοπικές μεταβλητές

variables.c

```
int total_calls = 0;

int next_of(int x)
{
  x = x + 1;
  total_calls = total_calls + 1;
  return x;
}

int main()
{
  int x = 0;
  x = next_of(0);
  x = next_of(x);
  return 0;
}
```

total_calls = 1,
x (main) = 1

# Παράδειγμα: Καθολικές και τοπικές μεταβλητές

variables.c

```
int total_calls = 0;

int next_of(int x)
{
  x = x + 1;
  total_calls = total_calls + 1;
  return x;
}

int main()
{
  int x = 0;
  x = next_of(0);
  x = next_of(x);
  return 0;
}
```

total_calls = 1,
x (main) = 1,
x (next_of) = 1

# Παράδειγμα: Καθολικές και τοπικές μεταβλητές

variables.c

```
int total_calls = 0;

int next_of(int x)
{
  x = x + 1;
  total_calls = total_calls + 1;
  return x;
}

int main()
{
  int x = 0;
  x = next_of(0);
  x = next_of(x);
  return 0;
}
```

total_calls = 1,
x (main) = 1,
x (next_of) = 1

# Παράδειγμα: Καθολικές και τοπικές μεταβλητές

variables.c

```
int total_calls = 0;

int next_of(int x)
{
  x = x + 1;
  total_calls = total_calls + 1;
  return x;
}

int main()
{
  int x = 0;
  x = next_of(0);
  x = next_of(x);
  return 0;
}
```

total_calls = 1,
x (main) = 1,
x (next_of) = 2

# Παράδειγμα: Καθολικές και τοπικές μεταβλητές

variables.c

```
int total_calls = 0;

int next_of(int x)
{
  x = x + 1;
  total_calls = total_calls + 1;
  return x;
}

int main()
{
  int x = 0;
  x = next_of(0);
  x = next_of(x);
  return 0;
}
```

total_calls = 2,
x (main) = 1,
x (next_of) = 2

# Παράδειγμα: Καθολικές και τοπικές μεταβλητές

variables.c

```
int total_calls = 0;

int next_of(int x)
{
  x = x + 1;
  total_calls = total_calls + 1;
  return x;
}

int main()
{
  int x = 0;
  x = next_of(0);
  x = next_of(x);
  return 0;
}
```

total_calls = 2,
x (main) = 1,
x (next_of) = 2

# Παράδειγμα: Καθολικές και τοπικές μεταβλητές

variables.c

```
int total_calls = 0;

int next_of(int x)
{
  x = x + 1;
  total_calls = total_calls + 1;
  return x;
}

int main()
{
  int x = 0;
  x = next_of(0);
  x = next_of(x);
  return 0;
}
```

total_calls = 2,
x (main) = 2

# Στατικές (static) μεταβλητές

- Ανάλογα με το είδος της εμβέλειας
  - Στατικές τοπικές μεταβλητές:
    - Δημιουργούνται και αρχικοποιούνται στην αρχή της εκτέλεσης
    - Έχουν τοπική εμβέλεια: μόνο στη συνάρτηση που τις δηλώνει
    - Όπως οι τοπικές, αλλά δεν χάνονται στο τέλος της συνάρτησης
    - Η συνάρτηση "θυμάται" την τιμή της μεταβλητής μεταξύ διαφορετικών κλήσεων
  - Στατικές καθολικές μεταβλητές:
    - Δημιουργούνται και αρχικοποιούνται στην αρχή της εκτέλεσης
    - Έχουν καθολική εμβέλεια: ορατές από όλες τις συναρτήσεις στο αρχείο
    - Δεν είναι ορατές από συναρτήσεις που δηλώνονται σε άλλα αρχεία
    - Όταν ένα μικρό σύνολο από συναρτήσεις χρησιμοποιούν την ίδια καθολική μεταβλητή, την κάνουμε στατική καθολική και μπαίνουν αυτές οι συναρτήσεις στο ίδιο αρχείο

# Παράδειγμα

## staticvar.c

```
void f()
{
  static int static_var = 500;
  printf("%d\n", static_var);
  static_var++;
}

int main()
{
  f();
  f();
  f();
}
```

# Παράδειγμα

Η συνάρτηση δεν έχει ορίσματα και δεν επιστρέφει αποτέλεσμα

staticvar.c

```
void f()
{
  static int static_var = 500;
  printf("%d\n", static_var);
  static_var++;
}

int main()
{
  f();
  f();
  f();
}
```

# Παράδειγμα

staticvar.c

```
void f()
{
  static int static_var = 500;
  printf(”%d\n”, static_var);
  static_var++;
}

int main()
{
  f();
  f();
  f();
}
```

Δήλωση της στατικής τοπικής μεταβλητής, στην αρχή του προγράμματος έχει την τιμή 500

# Παράδειγμα

staticvar.c

```
void f()
{
  static int static_var = 500;
  printf("%d\n", static_var);
  static_var++;
}

int main()
{
  f();
  f();
  f();
}
```

Τυπώνει την τιμή της static_var

# Παράδειγμα

## staticvar.c

```
void f()
{
  static int static_var = 500;
  printf("%d\n", static_var);
  static_var++;
}

int main()
{
  f();
  f();
  f();
}
```

Αυξάνει την static_var κατά 1

# Παράδειγμα

## staticvar.c

```
void f()
{
  static int static_var = 500;
  printf("%d\n", static_var);
  static_var++;
}

int main()
{
  f();
  f();
  f();
}
```

Θα τυπώσει “500”

# Παράδειγμα

## staticvar.c

```
void f()
{
  static int static_var = 500;
  printf("%d\n", static_var);
  static_var++;
}

int main()
{
  f();
  f();
  f();
}
```

Θα τυπώσει “501”

# Παράδειγμα

## staticvar.c

```
void f()
{
  static int static_var = 500;
  printf("%d\n", static_var);
  static_var++;
}

int main()
{
  f();
  f();
  f();
}
```

Θα τυπώσει “502”

# Τύποι μεταβλητών

- Όλες οι μεταβλητές έχουν ένα τύπο δεδομένων
- Βασικοί τύποι:
  - `char`: Ακέραιος αριθμός μεγέθους 1 χαρακτήρα, συνήθως 1 byte (8 bits)
  - `int`: Ακέραιος αριθμός μεγέθους 1 λέξης (ορίζεται ανάλογα με τη μηχανή)
  - `float`: Αριθμός κινητής υποδιαστολής, απλής ακρίβειας, μεγέθους 32 bits
  - `double`: Αριθμός κινητής υποδιαστολής, διπλής ακρίβειας, μεγέθους 64 bits
- Επιπλέον προσδιορισμοί σε κάποιους τύπους
  - `signed`, `unsigned`: Προσδιορίζει αν ένας τύπος ακεραίων (char, int) αναπαριστά προσημασμένους αριθμούς ή όχι
  - `short`, `long`, `long long`: Προσδιορισμός μεγέθους ακεραίων (int), ορίζει το εύρος τιμών και το χώρο στη μνήμη
- Σε κάθε σημείο που χρησιμοποιείται μια μεταβλητή έχει τον τύπο με τον οποίο δηλώθηκε

# Αναπαράσταση και εύρος τιμών

- Ο τύπος `char`, έχει μέγεθος 8 bits: αναπαριστά $2^8$ διαφορετικές τιμές, συνήθως ASCII χαρακτήρες
- Ο τύπος `int` αναπαριστά ακέραιους αριθμούς, διαλέγουμε μέγεθος ανάλογα με το μέγιστο αριθμό που χρειαζόμαστε
- Οι τύποι `float`, `double` αναπαριστούν πραγματικούς αριθμούς
- Σε τυπικό 64-bit x86 επεξεργαστή:

| Τύπος | Εύρος | bytes |
|---|---|---|
| `char` | 0 έως +255 ή -127 έως +127 | 1 |
| `signed char` | -127 έως +127 | 1 |
| `unsigned char` | 0 έως +255 | 1 |
| `signed short int` | $-(2^{15}-1)$ έως $2^{15}-1$ | 2 |
| `unsigned short int` | 0 έως $+2^{16}-1$ | 2 |
| `signed int` | $-(2^{31}-1)$ έως $+2^{31}-1$ | 4 |
| `unsigned int` | 0 έως $+2^{32}-1$ | 4 |
| `signed long int` | $-(2^{63}-1)$ έως $+2^{63}-1$ | 8 |
| `unsigned long int` | 0 έως $+2^{64}-1$ | 8 |
| `signed long long int` | $-(2^{63}-1)$ έως $+2^{63}-1$ | 8 |
| `unsigned long long int` | 0 έως $+2^{64}-1$ | 8 |
| `float` | -3.2$\times 10^{38}$ έως +3.2$\times 10^{38}$ | 4 |
| `double` | -1.7$\times 10^{308}$ έως +1.7$\times 10^{308}$ | 8 |

# Αναπαράσταση και εύρος τιμών

- Ο τύπος char, έχει μέγεθος 8 bits: αναπαριστά $2^8$ διαφορετικές τιμές, συνήθως ASCII χαρακτήρες
- Ο τύπος int αναπαριστά ακέραιους αριθμούς, διαλέγουμε μέγεθος ανάλογα με το μέγιστο αριθμό που χρειαζόμαστε
- Οι τύποι float, double αναπαριστούν πραγματικούς αριθμούς
- Σε τυπικό 64-bit x86 επεξεργαστή:

| Τύπος | Εύρος | bytes |
|---|---|---|
| char | 0 έως +255 ή -127 έως +127 | 1 |
| signed char | -127 έως +127 | 1 |
| unsigned char | 0 έως +255 | 1 |
| signed short int | $-(2^{15}-1)$ έως $2^{15}-1$ | 2 |
| unsigned short int | 0 έως $+2^{16}-1$ | 2 |
| signed int | $-(2^{31}-1)$ έως $+2^{31}-1$ | 4 |
| unsigned int | 0 έως $+2^{32}-1$ | 4 |
| signed long int | $-(2^{63}-1)$ έως $+2^{63}-1$ | 8 |
| unsigned long int | 0 έως $+2^{64}-1$ | 8 |
| signed long long int | $-(2^{63}-1)$ έως $+2^{63}-1$ | 8 |
| unsigned long long int | 0 έως $+2^{64}-1$ | 8 |
| float | -3.2$\times 10^{38}$ έως +3.2$\times 10^{38}$ | 4 |
| double | -1.7$\times 10^{308}$ έως +1.7$\times 10^{308}$ | 8 |

32767

# Αναπαράσταση και εύρος τιμών

- Ο τύπος `char`, έχει μέγεθος 8 bits: αναπαριστά $2^8$ διαφορετικές τιμές, συνήθως ASCII χαρακτήρες
- Ο τύπος `int` αναπαριστά ακέραιους αριθμούς, διαλέγουμε μέγεθος ανάλογα με το μέγιστο αριθμό που χρειαζόμαστε
- Οι τύποι `float`, `double` αναπαριστούν πραγματικούς αριθμούς
- Σε τυπικό 64-bit x86 επεξεργαστή:

| Τύπος | Εύρος | bytes |
|---|---|---|
| `char` | 0 έως +255 ή -127 έως +127 | 1 |
| `signed char` | -127 έως +127 | 1 |
| `unsigned char` | 0 έως +255 | 1 |
| `signed short int` | $-(2^{15}-1)$ έως $2^{15}-1$ | 2 |
| `unsigned short int` | 0 έως $+2^{16}-1$ | 2 |
| `signed int` | $-(2^{31}-1)$ έως $+2^{31}-1$ | 4 |
| `unsigned int` | 0 έως $+2^{32}-1$ | 4 |
| `signed long int` | $-(2^{63}-1)$ έως $+2^{63}-1$ | 8 |
| `unsigned long int` | 0 έως $+2^{64}-1$ | 8 |
| `signed long long int` | $-(2^{63}-1)$ έως $+2^{63}-1$ | 8 |
| `unsigned long long int` | 0 έως $+2^{64}-1$ | 8 |
| `float` | -3.2$\times 10^{38}$ έως +3.2$\times 10^{38}$ | 4 |
| `double` | -1.7$\times 10^{308}$ έως +1.7$\times 10^{308}$ | 8 |

65535

# Αναπαράσταση και εύρος τιμών

- Ο τύπος `char`, έχει μέγεθος 8 bits: αναπαριστά $2^8$ διαφορετικές τιμές, συνήθως ASCII χαρακτήρες
- Ο τύπος `int` αναπαριστά ακέραιους αριθμούς, διαλέγουμε μέγεθος ανάλογα με το μέγιστο αριθμό που χρειαζόμαστε
- Οι τύποι `float`, `double` αναπαριστούν πραγματικούς αριθμούς
- Σε τυπικό 64-bit x86 επεξεργαστή:

| Τύπος | Εύρος | bytes |
|---|---|---|
| `char` | 0 έως +255 ή -127 έως +127 | 1 |
| `signed char` | -127 έως +127 | 1 |
| `unsigned char` | 0 έως +255 | 1 |
| `signed short int` | $-(2^{15}-1)$ έως $2^{15}-1$ | 2 |
| `unsigned short int` | 0 έως $+2^{16}-1$ | 2 |
| `signed int` | $-(2^{31}-1)$ έως $+2^{31}-1$ | 4 |
| `unsigned int` | 0 έως $+2^{32}-1$ | 4 |
| `signed long int` | $-(2^{63}-1)$ έως $+2^{63}-1$ | 8 |
| `unsigned long int` | 0 έως $+2^{64}-1$ | 8 |
| `signed long long int` | $-(2^{63}-1)$ έως $+2^{63}-1$ | 8 |
| `unsigned long long int` | 0 έως $+2^{64}-1$ | 8 |
| `float` | -3.2$\times 10^{38}$ έως +3.2$\times 10^{38}$ | 4 |
| `double` | -1.7$\times 10^{308}$ έως +1.7$\times 10^{308}$ | 8 |

2147483647

# Αναπαράσταση και εύρος τιμών

- Ο τύπος `char`, έχει μέγεθος 8 bits: αναπαριστά $2^8$ διαφορετικές τιμές, συνήθως ASCII χαρακτήρες
- Ο τύπος `int` αναπαριστά ακέραιους αριθμούς, διαλέγουμε μέγεθος ανάλογα με το μέγιστο αριθμό που χρειαζόμαστε
- Οι τύποι `float`, `double` αναπαριστούν πραγματικούς αριθμούς
- Σε τυπικό 64-bit x86 επεξεργαστή:

| Τύπος | Εύρος | bytes |
|---|---|---|
| `char` | 0 έως +255 ή -127 έως +127 | 1 |
| `signed char` | -127 έως +127 | 1 |
| `unsigned char` | 0 έως +255 | 1 |
| `signed short int` | $-(2^{15}-1)$ έως $2^{15}-1$ | 2 |
| `unsigned short int` | 0 έως $+2^{16}-1$ | 2 |
| `signed int` | $-(2^{31}-1)$ έως $+2^{31}-1$ | 4 |
| `unsigned int` | 0 έως $+2^{32}-1$ | 4 |
| `signed long int` | $-(2^{63}-1)$ έως $+2^{63}-1$ | 8 |
| `unsigned long int` | 0 έως $+2^{64}-1$ | 8 |
| `signed long long int` | $-(2^{63}-1)$ έως $+2^{63}-1$ | 8 |
| `unsigned long long int` | 0 έως $+2^{64}-1$ | 8 |
| `float` | -3.2$\times 10^{38}$ έως +3.2$\times 10^{38}$ | 4 |
| `double` | -1.7$\times 10^{308}$ έως +1.7$\times 10^{308}$ | 8 |

4294967295

# Αναπαράσταση και εύρος τιμών

- Ο τύπος `char`, έχει μέγεθος 8 bits: αναπαριστά $2^8$ διαφορετικές τιμές, συνήθως ASCII χαρακτήρες
- Ο τύπος `int` αναπαριστά ακέραιους αριθμούς, διαλέγουμε μέγεθος ανάλογα με το μέγιστο αριθμό που χρειαζόμαστε
- Οι τύποι `float`, `double` αναπαριστούν πραγματικούς αριθμούς
- Σε τυπικό 64-bit x86 επεξεργαστή:

| Τύπος | Εύρος | bytes |
|---|---|---|
| `char` | 0 έως +255 ή -127 έως +127 | 1 |
| `signed char` | -127 έως +127 | 1 |
| `unsigned char` | 0 έως +255 | 1 |
| `signed short int` | $-(2^{15}-1)$ έως $2^{15}-1$ | 2 |
| `unsigned short int` | 0 έως $+2^{16}-1$ | 2 |
| `signed int` | $-(2^{31}-1)$ έως $+2^{31}-1$ | 4 |
| `unsigned int` | 0 έως $+2^{32}-1$ | 4 |
| `signed long int` | $-(2^{63}-1)$ έως $+2^{63}-1$ | 8 |
| `unsigned long int` | 0 έως $+2^{64}-1$ | 8 |
| `signed long long int` | $-(2^{63}-1)$ έως $+2^{63}-1$ | 8 |
| `unsigned long long int` | 0 έως $+2^{64}-1$ | 8 |
| `float` | -3.2$\times 10^{38}$ έως +3.2$\times 10^{38}$ | 4 |
| `double` | -1.7$\times 10^{308}$ έως +1.7$\times 10^{308}$ | 8 |

9223372036854775807

# Αναπαράσταση και εύρος τιμών

- Ο τύπος `char`, έχει μέγεθος 8 bits: αναπαριστά $2^8$ διαφορετικές τιμές, συνήθως ASCII χαρακτήρες
- Ο τύπος `int` αναπαριστά ακέραιους αριθμούς, διαλέγουμε μέγεθος ανάλογα με το μέγιστο αριθμό που χρειαζόμαστε
- Οι τύποι `float`, `double` αναπαριστούν πραγματικούς αριθμούς
- Σε τυπικό 64-bit x86 επεξεργαστή:

| Τύπος | Εύρος | bytes |
|---|---|---|
| `char` | 0 έως +255 ή -127 έως +127 | 1 |
| `signed char` | -127 έως +127 | 1 |
| `unsigned char` | 0 έως +255 | 1 |
| `signed short int` | $-(2^{15}-1)$ έως $2^{15}-1$ | 2 |
| `unsigned short int` | 0 έως $+2^{16}-1$ | 2 |
| `signed int` | $-(2^{31}-1)$ έως $+2^{31}-1$ | 4 |
| `unsigned int` | 0 έως $+2^{32}-1$ | 4 |
| `signed long int` | $-(2^{63}-1)$ έως $+2^{63}-1$ | 8 |
| `unsigned long int` | 0 έως $+2^{64}-1$ | 8 |
| `signed long long int` | $-(2^{63}-1)$ έως $+2^{63}-1$ | 8 |
| `unsigned long long int` | 0 έως $+2^{64}-1$ | 8 |
| `float` | -3.2$\times 10^{38}$ έως +3.2$\times 10^{38}$ | 4 |
| `double` | -1.7$\times 10^{308}$ έως +1.7$\times 10^{308}$ | 8 |

18446744073709551615

# Τύποι σταθερών

- Ο τύπος μιας σταθεράς εξαρτάται από τον τρόπο που εκφράζεται
  - Σταθερές χαρακτήρων
    - `'x'` : σταθερά τύπου `int` (προσοχή: όχι `char`) με τιμή τον αντίστοιχο ακέραιο ASCII
  - Ακέραιες σταθερές
    - `42` : σταθερά τύπου `int`
    - `42U` : σταθερά τύπου `unsigned int`
    - `42L` : σταθερά τύπου `long int`
  - Σταθερές κινητής υποδιαστολής
    - `3.14159` : σταθερά τύπου `double`
    - `3.14159f` : σταθερά τύπου `float`

# Τελεστές και προτεραιότητα

1. Παρένθεση ( )
2. Άρνηση !
3. Πολλαπλασιασμός *, διαίρεση /, υπόλοιπο %
4. Πρόσθεση +, αφαίρεση -
5. Ανισότητες <, >, <=, >=
6. Ισότητα ==, μη ισότητα !=,
7. Λογικό “και” &&
8. Λογικό “ή” ||

- Όταν δεν είστε σίγουροι, χρησιμοποιήστε παρενθέσεις

# Εκφράσεις

- Έκφραση (expression)
  - Στα ελληνικά: μια σειρά από λέξεις που έχει ένα νόημα
  - Στη C: μια σειρά από "λέξεις" που περιγράφει έναν απλό υπολογισμό
  - Το αποτέλεσμα του υπολογισμού είναι η *τιμή* της έκφρασης
- Οι μεταβλητές και οι σταθερές είναι "βασικές" εκφράσεις (λέξεις)
  - `x`, `42`, `"Hello, world!"`
- Οι τελεστές συνδυάζουν μικρότερες εκφράσεις σε μεγαλύτερες
  - `x + y`, `(x != 2) && (y - 5 + z * 3 > x)`, `'a' + 2`

# Τύπος έκφρασης

- Κάθε έκφραση υπολογίζει μια τιμή
- Κάθε έκφραση έχει ένα τύπο
  - Η έκφραση `1 + 1` έχει τύπο `int` και τιμή 2
  - Η έκφραση `5/2` έχει τύπο `int` και τιμή 2
  - Η έκφραση `5.0/2` εχει τύπο `double` και τιμή 2.5
  - Η έκφραση `1 == 0` έχει τύπο `int` και τιμή 0
- Οι τελεστές έχουν τον "μεγαλύτερο" από τους τύπους των τελεσταίων
- Οι λογικοί τελεστές επιστρέφουν 1 για "αληθές" ή 0 για "ψευδές"

# Μετατροπές τύπων

- Τί γινεται αν αποθηκεύσουμε μια τιμή ενός τύπου σε μεταβλητή άλλου τύπου;
- Κάποιες μετατροπές γίνονται αυτόματα, π.χ: `int` $\rightarrow$ `double`
- Κάποιοι τύποι "χωράνε" ακριβώς σε άλλους, π.χ: `char` $\rightarrow$ `int`
- Πρόβλημα: η μετατροπή μπορεί να χάνει δεδομένα!
  - `int x = 15.0/2.0;`
  - `char x = 256;`
  - Ο τύπος `char` μπορεί να είναι `unsigned` ή `signed` ανάλογα με το μεταφραστή

# Ρητή μετατροπή τύπων

- Type casting: η ρητή μετατροπή δεδομένων από την αναπαράσταση ενός τύπου σε αναπαράσταση άλλου τύπου
  - Π.χ: αν το y είναι `char`
  - `x = (int) y`
- Γενική μορφή
  - `έκφραση1 = (τύπος) έκφραση2`
  - Η τιμή της `έκφραση2` μετατρέπεται στην αναπαράσταση τύπος και αποθηκεύεται στην `έκφραση1`
- Προσοχή: επιτρέπεται ακόμα και όταν δεν έχει νόημα!
  - `x = (unsigned int) -1`
  - Συχνή αιτία λαθών σε προγράμματα C

# Τελεστές αύξησης-μείωσης

- Τελεστής `++`
  - Η μεταβλητή `i` αυξάνεται κατά 1
- Τελεστής `--`
  - Η μεταβλητή `i` μειώνεται κατά 1
- Μεταθεματικοί (postfix): ο τελεστής μετά τον τελεσταίο
  - Η έκφραση `i++` έχει την τιμή του `i` πριν την αύξηση
- Προθεματικοί (prefix): ο τελεστής πριν τον τελεσταίο
  - Η έκφραση `++i` έχει την τελευταία (αυξημένη) τιμή του `i`
- Αν δεν χρησιμοποιείται η τιμή της έκφρασης είναι ισοδύναμα
- Προτεραιότητα όπως η πρόσθεση

# Τελεστές αντικατάστασης

- Αύξηση: `x += y;`
  - Ισοδύναμο με `x = x + y;`
- Μείωση: `x -= y;`
- Πολλαπλασιασμός: `x *= y;`
- Διαίρεση: `x /= y;`
- κλπ ...
- Γενικά: `έκφραση1 τελεστής= έκφραση2`
- Υπολογίζονται πρώτα οι `έκφραση1`, `έκφραση2`
- *Μετά*, εφαρμόζεται ο `τελεστής` στις τιμές τους
- Το αποτέλεσμα αποθηκεύεται στην θέση της `έκφραση1`
  - Π.χ: το `x *= y + 2;` είναι ισοδύναμο με `x = x * (y + 2);`

# Είσοδος - Έξοδος

- `printf` συνάρτηση εξόδου
- `scanf` συνάρτηση εισόδου
- Η πρώτη παράμετρος είναι η *συμβολοσειρά μορφής* (format string) και περιγράφει πόσες παράμετροι ακολουθούν, και πως θα εκτυπωθούν
- Ακολουθούν οι παράμετροι που περιγράφονται στη συμβολοσειρά μορφής, πρέπει να είναι του σωστού τύπου
- Παράμετροι
  - `%c` — Χαρακτήρας (char)
  - `%d` — Ακέραιος σε δεκαδικό (signed int)
  - `%X` — Ακέραιος σε δεκαεξαδικό (unsigned int)
  - `%f` — Κινητής υποδιαστολής (float)
  - `%lf` — Κινητής υποδιαστολής, διπλής ακρίβειας (double)
  - Για περισσότερα: "`man 3 printf`"

# Παραδείγματα

## io1.c

```
#include<stdio.h>
int main() {
  int x;
  char c;
  scanf("%d", &x);
  c = 'a';
  printf("x = %d\n", x);
  printf("c(char) = %c\n", c);
  printf("c(int) = %d\n", c);
}
```

# Παραδείγματα

## io1.c

```
#include<stdio.h>
int main() {
  int x;
  char c;
  scanf("%d", &x);
  c = 'a';
  printf("x = %d\n", x);
  printf("c(char) = %c\n", c);
  printf("c(int) = %d\n", c);
}
```

Περιμένει έναν δεκαδικό ακέραιο από την είσοδο

# Παραδείγματα

## io1.c

```
#include<stdio.h>
int main() {
  int x;
  char c;
  scanf("%d", &x);
  c = 'a';
  printf("x = %d\n", x);
  printf("c(char) = %c\n", c);
  printf("c(int) = %d\n", c);
}
```

Αποθηκεύει την τιμή που διάβασε στο x

# Παραδείγματα

## io1.c

```
#include<stdio.h>
int main() {
  int x;
  char c;
  scanf("%d", &x);
  c = 'a';
  printf("x = %d\n", x);
  printf("c(char) = %c\n", c);
  printf("c(int) = %d\n", c);
}
```

Η μεταβλητή c περιέχει την τιμή ASCII του συμβόλου a, 97

# Παραδείγματα

## io1.c

```
#include<stdio.h>
int main() {
  int x;
  char c;
  scanf("%d", &x);
  c = 'a';
  printf("x = %d\n", x);
  printf("c(char) = %c\n", c);
  printf("c(int) = %d\n", c);
}
```

Το επόμενο όρισμα της `printf` μετά το format string να εκτυπωθεί ως δεκαδικός ακέραιος

# Παραδείγματα

## io1.c

```
#include<stdio.h>
int main() {
  int x;
  char c;
  scanf("%d", &x);
  c = 'a';
  printf("x = %d\n", x);
  printf("c(char) = %c\n", c);
  printf("c(int) = %d\n", c);
}
```

Περιέχει τον αριθμό που διαβάστηκε από την είσοδο, αντιστοιχεί στο %d

# Παραδείγματα

## io1.c

```
#include<stdio.h>
int main() {
  int x;
  char c;
  scanf("%d", &x);
  c = 'a';
  printf("x = %d\n", x);
  printf("c(char) = %c\n", c);
  printf("c(int) = %d\n", c);
}
```

Το επόμενο όρισμα να εκτυπωθεί ως χαρακτήρας

# Παραδείγματα

## io1.c

```
#include<stdio.h>
int main() {
  int x;
  char c;
  scanf("%d", &x);
  c = 'a';
  printf("x = %d\n", x);
  printf("c(char) = %c\n", c);
  printf("c(int) = %d\n", c);
}
```

Περιέχει την τιμή 97, που αντιστοιχεί στο "a"

# Παραδείγματα

## io1.c

```
#include<stdio.h>
int main() {
  int x;
  char c;
  scanf("%d", &x);
  c = 'a';
  printf("x = %d\n", x);
  printf("c(char) = %c\n", c);
  printf("c(int) = %d\n", c);
}
```

Το επόμενο όρισμα να εκτυπωθεί ως δεκαδικός ακέραιος

# Παραδείγματα

## io1.c

```
#include<stdio.h>
int main() {
  int x;
  char c;
  scanf("%d", &x);
  c = 'a';
  printf("x = %d\n", x);
  printf("c(char) = %c\n", c);
  printf("c(int) = %d\n", c);
}
```

Περιέχει την τιμή 97, που αντιστοιχεί στο "a"

# Παραδείγματα

## io1.c

```
#include<stdio.h>
int main() {
  int x;
  char c;
  scanf("%d", &x);
  c = 'a';
  printf("x = %d\n", x);
  printf("c(char) = %c\n", c);
  printf("c(int) = %d\n", c);
}
```

# Παραδείγματα

## io1.c

```
#include<stdio.h>
int main() {
  int x;
  char c;
  scanf("%d", &x);
  c = 'a';
  printf("x = %d\n", x);
  printf("c(char) = %c\n", c);
  printf("c(int) = %d\n", c);
}
```

## io2.c

```
#include<stdio.h>
int main() {
  float x;
  scanf("%f", &x);
  printf("x = %f\n, x);
  printf("x ~= %.2f\n", x);
}
```

# Παραδείγματα

## io1.c

```
#include<stdio.h>
int main() {
  int x;
  char c;
  scanf("%d", &x);
  c = 'a';
  printf("x = %d\n", x);
  printf("c(char) = %c\n", c);
  printf("c(int) = %d\n", c);
}
```

## io2.c

```
#include<stdio.h>
int main() {
  float x;
  scanf("%f", &x);
  printf("x = %f\n, x);
  printf("x ~= %.2f\n", x);
}
```

Περιμένει έναν δεκαδικό αριθμό με υποδιαστολή από την είσοδο

# Παραδείγματα

## io1.c

```
#include<stdio.h>
int main() {
  int x;
  char c;
  scanf("%d", &x);
  c = 'a';
  printf("x = %d\n", x);
  printf("c(char) = %c\n", c);
  printf("c(int) = %d\n", c);
}
```

## io2.c

```
#include<stdio.h>
int main() {
  float x;
  scanf("%f", &x);
  printf("x = %f\n, x);
  printf("x ~= %.2f\n", x);
}
```

Αποθηκεύει την τιμή που διάβασε στο x ως αριθμό κινητής υποδιαστολής

# Παραδείγματα

## io1.c

```
#include<stdio.h>
int main() {
  int x;
  char c;
  scanf("%d", &x);
  c = 'a';
  printf("x = %d\n", x);
  printf("c(char) = %c\n", c);
  printf("c(int) = %d\n", c);
}
```

## io2.c

```
#include<stdio.h>
int main() {
  float x;
  scanf("%f", &x);
  printf("x = %f\n", x);
  printf("x ~= %.2f\n", x);
}
```

Το όρισμα της `printf` μετά το format string να εκτυπωθεί ως αριθμός κινητής υποδιαστολής

# Παραδείγματα

## io1.c

```
#include<stdio.h>
int main() {
  int x;
  char c;
  scanf("%d", &x);
  c = 'a';
  printf("x = %d\n", x);
  printf("c(char) = %c\n", c);
  printf("c(int) = %d\n", c);
}
```

## io2.c

```
#include<stdio.h>
int main() {
  float x;
  scanf("%f", &x);
  printf("x = %f\n, x);
  printf("x ~= %.2f\n", x);
}
```

Το επόμενο όρισμα να εκτυπωθεί με 2 ακριβώς ψηφία μετά την υποδιαστολή

# Παραδείγματα

## io1.c

```
#include<stdio.h>
int main() {
  int x;
  char c;
  scanf("%d", &x);
  c = 'a';
  printf("x = %d\n", x);
  printf("c(char) = %c\n", c);
  printf("c(int) = %d\n", c);
}
```

## io2.c

```
#include<stdio.h>
int main() {
  float x;
  scanf("%f", &x);
  printf("x = %f\n, x);
  printf("x ~= %.2f\n", x);
}
```

# Παραδείγματα

## io1.c

```
#include<stdio.h>
int main() {
  int x;
  char c;
  scanf("%d", &x);
  c = 'a';
  printf("x = %d\n", x);
  printf("c(char) = %c\n", c);
  printf("c(int) = %d\n", c);
}
```

## io2.c

```
#include<stdio.h>
int main() {
  float x;
  scanf("%f", &x);
  printf("x = %f\n, x);
  printf("x ~= %.2f\n", x);
}
```

# Συχνά λάθη

- `printf`
  - Λιγότερα ορίσματα από όσα περιγράφει το format string
  - Διαφορετικά ορίσματα από αυτά που περιγράφει το format string
- `scanf`
  - Όλα τα παραπάνω
  - Ξεχασμένο &
- Λάθη σε μετατροπές τύπων
  - Αυτόματες μετατροπές, αναθέσεις μεταξύ διαφορετικών τύπων, κλπ
- Λάθος τελεστές
  - Π.χ., ανάθεση αντί ισότητας
  - `if (x = 5)` αντί `if (x == 5)`
- Ξεχασμένο semicolon “;”
  - Ή semicolon σε θέση όπου δεν χρειάζεται

# Τί θα τυπώσει;

## example1.c

```
#include <stdio.h>
int main()
{
  int a = 1, b = 1, x = 0, y = 0;
  double w;

  x = 1 + a++;
  printf("x = %d\n", x);
  printf("a = %d\n", a);
  y = ++b;
  printf("y = %d\n", y);
  printf("b = %d\n", b);
}
```

# Τί θα τυπώσει;

## example1.c

```
#include <stdio.h>
int main()
{
  int a = 1, b = 1, x = 0, y = 0;
  double w;

  x = 1 + a++;
  printf("x = %d\n", x);
  printf("a = %d\n", a);
  y = ++b;
  printf("y = %d\n", y);
  printf("b = %d\n", b);
}
```

## Output

```
x = 2
a = 2
y = 2
b = 2
```