ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΕΣ ΤΕΧΝΙΚΕΣ

http://www.softlab.ntua.gr/~nickie/Courses/progtech/

Διδάσκοντες: Γιάννης Μαΐστρος (maistros@cs.ntua.gr)
Στάθης Ζάχος (zachos@cs.ntua.gr)
Νίκος Παπασπύρου (nickie@softlab.ntua.gr)

Διαφάνειες παρουσίασης #9 (α)

- ✓Δείκτες σε συναρτήσεις
- ✓Υλοποίηση αριθμητικών εκφράσεων

Δείκτες σε συναρτήσεις

- Παράδειγμα 1

```
int f (int n);
int (*p) (int n);

p = &f;
printf("%d", (*p)(3));
```

- Παράδειγμα 2

```
double (*q) (double) = &sqrt;
```

- Δείκτες ισοδύναμοι με συναρτήσεις

```
p = f;
printf("%d", p(3));
```

Ολοκλήρωση

```
double integral (double a, double b,
                 double (*f) (double))
{
   const double step = 1e-6;

   double result = 0.0;
   double x;

   for (x=a; x<=b; x += step)
      result += f(x) * step;
   return result;
}

...

result = integral(0, 3.14159, sin);
```

κακός αλγόριθμος!

Ταξινόμηση φυσαλίδας (i)

- Τα δεδομένα:
  - `void * a` πού βρίσκονται
  - `int n` πόσα είναι
  - `int size` τί μέγεθος έχει καθένα
  - `comparison f` πώς γίνεται η σύγκριση
- Η σύγκριση των δεδομένων

```
typedef int (*comparison) (void * x,
                           void * y);
```

Ταξινόμηση φυσαλίδας (ii)

```
void bsort (void * a, int n,
            comparison f, int size)
{
  int i, j;

  for (i = 0; i < n; i++)
    for (j = n-1; j > i; j--) {
      unsigned char * px =
        (unsigned char *) a +
        (j-1) * size;
      unsigned char * py =
        (unsigned char *) a +
        j * size;
```

Ταξινόμηση φυσαλίδας (iii)

- (συνέχεια)

```
            if (f(px, py) > 0) {
               int k;

               for (k = 0; k < size; k++) {
                  unsigned char temp = *px;

                  *px++ = *py;
                  *py++ = temp;
               }
            }
         }
      }
```

Ταξινόμηση φυσαλίδας (iv)

- Συνάρτηση σύγκρισης για ακέραιους

```
int intcompare (void * x, void * y)
{
   return *((int *) x) - *((int *) y);
}
```

- Κλήση για πίνακα ακεραίων

```
int x[] = { 44, 55, 12, 42, ... };
...
bsort(x, n, intcompare, sizeof(int));
```

Υλοποίηση αριθμητικών εκφράσεων

- Έκφραση: δυαδικό δένδρο ειδικής μορφής
- Κόμβοι τριών ειδών:
  - αριθμητικές σταθερές (φύλλα)
  - τελεστές με ένα τελούμενο
  - τελεστές με δύο τελούμενα
- Πληροφορία:
  - τιμή σταθεράς, ή
  - σύμβολο τελεστή
- Σύνδεσμοι:
  - τελούμενα

Κόμβος δένδρου

```
typedef struct ExprNode_tag {
  enum { EXP_const, EXP_unop,
         EXP_binop } code;
  union {
    double constant;
    struct {
      char op;
      struct ExprNode_tag *arg;
    } unop;
    struct {
      char op;
      struct ExprNode_tag *arg1, *arg2;
    } binop;
  } data;
} ExprNode, *expr;
```

Κατασκευή σταθερών

```
expr exprConst (double constant)
{
    ExprNode * e = (ExprNode *)
        malloc(sizeof(ExprNode));

    if (e == NULL) {
        printf("Out of memory\n");
        exit(1);
    }

    e->code = EXP_const;
    e->data.constant = constant;
    return e;
}
```

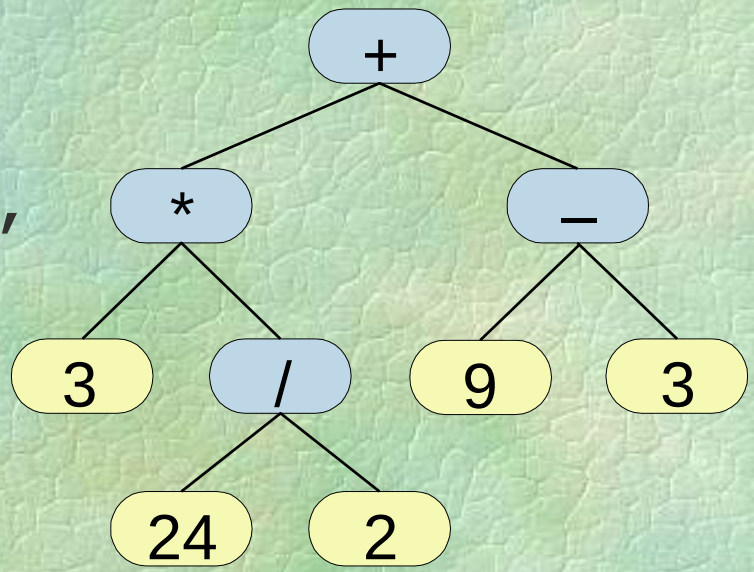
Κατασκευή τελεστών με δύο τελούμενα

```
expr exprBinop (expr arg1, char op,
                expr arg2)
{
    ExprNode * e = (ExprNode *)
       malloc(sizeof(ExprNode));

    if (e == NULL) {
       printf("Out of memory\n");
       exit(1);
    }

    e->code = EXP_binop;
    e->data.binop.op = op;
    e->data.binop.arg1 = arg1;
    e->data.binop.arg2 = arg2;
    return e;
}
```

Παράδειγμα κατασκευής

- Έκφραση: **3 * (24 / 2) + (9 - 3)**

```
expr e =
 exprBinop(
  exprBinop(
   exprConst(3),'*',
   exprBinop(
    exprConst(24), '/',
    exprConst(2)
   )
  ), '+',
  exprBinop(
   exprConst(9), '-',
   exprConst(3)
  )
 );
```



Καταστροφή έκφρασης

```
void exprFree (expr e)
{
   switch (e->code) {
      case EXP_unop:
         exprFree(e->data.unop.arg);
         break;
      case EXP_binop:
         exprFree(e->data.binop.arg1);
         exprFree(e->data.binop.arg2);
         break;
   }
   free(e);
}
```

Εκτύπωση έκφρασης (i)

```
void exprPrint (expr e)
{
  switch (e->code) {
    case EXP_const:
      printf("%lg", e->data.constant);
      break;
    case EXP_unop:
      printf("(%c", e->data.unop.op);
      exprPrint(e->data.unop.arg);
      printf(")");
      break;
```

Εκτύπωση έκφρασης (ii)

- (συνέχεια)

```
        case EXP_binop:
          printf("(");
          exprPrint(e->data.binop.arg1);
          printf("%c", e->data.binop.op);
          exprPrint(e->data.binop.arg2);
          printf(")");
          break;
    }
  }
```

Υπολογισμός έκφρασης (i)

```
double exprCalc (expr e)
{
  switch (e->code) {
    case EXP_const:
      return e->data.constant;
    case EXP_unop:
      switch (e->data.unop.op) {
        case '+':
          return exprCalc(e->data.
                                unop.arg);
        case '-':
          return - exprCalc(e->data.
                                  unop.arg);
      }
```

Υπολογισμός έκφρασης (ii)

- (συνέχεια)

```
case EXP_binop:
  switch (e->data.binop.op) {
    case '+':
      return exprCalc(e->data.
                      binop.arg1)
           + exprCalc(e->data.
                      binop.arg2);
    case '-':
      return exprCalc(e->data.
                      binop.arg1)
           - exprCalc(e->data.
                      binop.arg2);
    ...
```