HY255 Εργαστήριο Λογισμικού – L24

(a) Programming Style

(b) Comments

HY255 Εργαστήριο Λογισμικού
Άνοιξη 2021
Άγγελος Μπίλας

?

```
long h[4]; t() { h[3]-=h[3]/3000; setitimer(0,h,0); }
c,d,l,v[]={(int)t,0,2},w,s,I,K=0,i=276,j,k,q[276],Q[276],*n=q,*m,x=17,
f[]={7,-13,-12,1,8,-11,-12,-1,9,-1,1,12,3,-13,-12,-1,12,-1,11,1,15,-
1,13,1,18,-1,1,2,0,-12,-1,11,1,-12,1,13,10,-12,1,12,11,-12,-1,1,2,-
12,-1,12,13,-12,12,13,14,-11,-1,1,4,-13,-12,12,16,-11,-12,12,17,-
13,1,-1,5,-12,12,11,6,-12,12,24}; u() { for(i=11;++i<264;)
if((k=q[i])-Q[i]) { Q[i]=k; if(i-++I||i%12<1)
printf("\033[%d;%dH",(I=i)/12,i%12*2+28); printf( "\033[%dm  "+(K-
k?0:5),k); K=k; } Q[263]=c=getchar(); } G(b) { for(i=4;i--;)
if(q[i?b+n[i]:b]) return 0; return 1; } g(b) { for(i=4;i--
;q[i?x+n[i]:x]=b); } main(C,V,a) char* *V,*a; {
h[3]=1000000/(l=C>1?atoi(V[1]):2); for(a=C>2?V[2]:"jkl pq";i;i--)
*n++=i<25||i%12<2?7:0; srand(getpid()); system("stty cbreak -echo stop
u"); sigvec(14,v,0); t(); puts("\033[H\033[J");
for(n=f+rand()%7*4;;g(7),u(),g(0)) { if(c<0) { if(G(x+12)) x+=12; else
{ g(7);++w; for(j=0;j<252;j=12*(j/12+1)) for(;q[++j];) if(j%12==10) {
for(;j%12;q[j--]=0); u(); for(;--j;q[j+12]=q[j]); u(); }
n=f+rand()%7*4; G(x=17)||(c=a[5]); } } if(c==*a)G(--x)||++x;
if(c==a[1])n=f+4**(m=n),G(x)||(n=m); if(c==a[2])G(++x)||--x;
if(c==a[3]) for(;G(x+12);++w) x+=12; if(c==a[4]||c==a[5]) {
s=sigblock(8192); printf("\033[H\033[J\033[0m%d\n",w);
if(c==a[5])break; for(j=264;j--;Q[j]=0); while(getchar()-a[4]);
puts("\033[H\033[J\033[7m"); sigsetmask(s); } } d=popen("stty -cbreak
echo stop \023;cat - HI|sort -rn|head -20>/tmp/$$;mv /tmp/$$ HI\ ;cat
HI","w"); fprintf(d,"%4d on level %1d by %s\n",w,l,getlogin());
pclose(d); }
```

- Και όμως κάνει compile…

Unix, a Hoax?

- https://www.gnu.org/fun/jokes/unix-hoax.html **[look at the date]**
- [...] Then Dennis and Brian worked on a truly warped version of Pascal, called 'A'. When we found others were actually trying to create real programs with A, we quickly added additional cryptic features and evolved into B, BCPL and finally C. We stopped when we got a clean compile on the following syntax:

```
for(;P("\n"),R--;P("|"))for(e=C;e--;P("_"+(*u++/8)%2))P("| "+(*u/4)%2);
```

- To think that modern programmers would try to use a language that allowed such a statement was beyond our comprehension! We actually thought of selling this to the Soviets to set their computer science progress back 20 or more years.
- Imagine our surprise when AT&&T and other US corporations actually began trying to use Unix and C! It has taken them 20 years to develop enough expertise to generate even marginally useful applications using this 1960's technological parody, but we are impressed with the tenacity (if not common sense) of the general Unix and C programmer.
- In any event, Brian, Dennis and I have been working exclusively in Pascal on the Apple Macintosh for the past few years and feel really guilty about the chaos, confusion and truly bad programming that have resulted from our silly prank so long ago.” [...]

Code Reviews

- Πως εξασφαλίζουμε ότι ένα κομμάτι κώδικα είναι “υψηλής ποιότητας”
  - Τι θα μπορούσε να σημαίνει “υψηλής ποιότητας”;
    - (1) Να είναι “καλά” σχεδιασμένο
    - (2) Να μπορούμε να βρούμε σχετικά εύκολα λάθη/bugs
    - (3) Να μπορεί να το καταλάβει/αλλάξει αυτός που το έγραψε
    - (4) Να μπορεί να το καταλάβει/αλλάξει κάποιος τρίτος
  - Σε πραγματικά συστήματα, παίζουν ρόλο όλα και εξίσου
- Τι μπορούμε να κάνουμε;
- Testing → οκ, αλλά συνήθως μόνο για μέρος του (2)
  - Τα assertions είναι ένας άλλος σημαντικός μηχανισμός για debugging
  - Τα code reviews ένας άλλος εξίσου σημαντικός μηχανισμός, ακόμη και για debugging
- Code reviews: Ένα σημαντικό εργαλείο για σχεδιασμό, debugging, και maintenance
  - Code reviews = Διαβάζουμε/εξηγούμε τον κώδικα offline, καταλαβαίνουμε τι κάνει, και καταλήγουμε ότι συμφωνούμε ή διαφωνούμε με τον τρόπο που το κάνει
- Πως μπορούν τα code reviews να γίνουν ουσιαστικά;
- Ένα σημαντικό στοιχείο είναι να μπορεί κανείς να διαβάσει τον κώδικα και να τον καταλάβει – σίγουρα όχι:

```
for(;P("\n"),R--;P("|"))for(e=C;e--;P("_"+(*u++/8)%2))P("| "+(*u/4)%2);
```

- Πως γράφουμε αναγνώσιμο κώδικα; → programming style

Programming Style

- Λίγοι (ουκ εν το πολλώ το ευ) και απλοί κανόνες για
  - (Α) Ονόματα και αριθμοί
  - (Β) Εντολές
  - (Γ) Εκφράσεις
  - (Δ) Σχόλια
- Υπάρχουν διάφορες παραλλαγές που μπορεί να σας αρέσουν
  - Χρησιμοποιείτε πάντα την ίδια – be consistent
  - Συμφωνείστε με τα μέλη της (εκάστοτε) ομάδας σας και τους συνεργάτες σας ποια παραλλαγή θα χρησιμοποιείτε
- Χρησιμοποιείτε εργαλεία για να κάνετε enforce κάποιους από αυτούς τους κανόνες
  - Π.χ. indent, clang-format για θέματα formatting

(A) Ονόματα και αριθμοί

- (1) Πρέπει να
  - Δίνουν πληροφορία
  - Είναι ευκολομνημόνευτα
  - Εύκολα στην ανάγνωση
  - Συνεπή στο συμβολισμός
- (2) Χρησιμοποιείστε
  - Περιγραφικά ονόματα για global vars
  - Σύντομα ονόματα για local vars

```
for(theElementIndex=0; theElementIndex<numberOfElements;
    theElementIndex++){
        elementArray[theElementIndex] = theElementIndex;
}
vs.
for(i=0; i<nelems; i++){
        elems[i] = i;
}
```

- (3) Χρησιμοποιείτε πάντα κάποιες συμβάσεις (ενδεικτικά)
  - CAPITALS for constants
  - FirstCapital for globals
  - lowerCase for locals
  - Use letter i,p,c,f to indicate type, e.g. pnode, fheight

(A) Ονόματα και αριθμοί

- (4) Προσπαθείστε να είστε συνεπείς μεταξύ των μεταβλητών
  - Σχετικές μεταβλητές να έχουν ονόματα που δείχνουν την μεταξύ τους σχέση

```
struct Queue {
        int noOfItemsInQ, frontOfTheQueue, queueCapacity;
} queue;
queue.queueCapacity++;
```

vs.

```
struct Queue {
        int nitems, front, capacity;
} queue;
queue.capacity++;
```

- (5) Χρησιμοποιείτε ”ενεργά” ονόματα για συναρτήσεις

```
now = getTime();
putchar('a');
```

- (6) Να είστε ακριβείς

```
if(checkoctal(c))... /* does it return true or false if c is octal? */
vs.
if(isoctal(c))...
```

- (7) Μην χρησιμοποιείτε αριθμούς αλλά αντικαταστήστε τους με ονόματα

```
for(i=0; i<255; i++)
vs.
#define NCHARS 255
for(i=0;i<NCHARS;i++)
```

```
vs.
int const NCHARS=255;
for(i=0;i<NCHARS;i++)
```

```
vs.
enum {
        NCHARS=255,
        MAXCOLORS=100,
        ...
};
for(i=0;i<NCHARS;i++)
```

(B) Εντολές

- (1) Χρησιμοποιείτε στοίχιση που αντιστοιχεί στη δομή του προγράμματος

```
    for(n++;n<100;field[n++]='\0');
    *i='\0'; return('\n');
vs.
    for(n++; n<100; field[n++]='\0')
        ;
    *i='\0';
    return('\n');
vs.
    for(n++; n<100; n++){
        field[n]='\0';
    }
    *i='\0';
    return('\n');
```

- Ειδικά για τη μορφοποίηση των εντολών υπάρχουν διάφορες παραλλαγές που μπορεί να σας αρέσουν
  - Χρησιμοποιείτε πάντα την ίδια – be consistent
  - Συμφωνείστε με τα μέλη της (εκάστοτε) ομάδας σας και τους συνεργάτες σας ποια παραλλαγή θα χρησιμοποιείτε

(Γ) Εκφράσεις

- (1) Χρησιμοποιείτε κενά γύρω από τελεστές (+,*,<,…) με δύο arguments
- (2) Χρησιμοποιείτε την θετική μορφή των εκφράσεων

```
if (!(block_id < high) || !(block_id >= low))
vs.
if ((block_id >= high) || (block_id < low))
vs.
if ((block_id < low) || (block_id >= high))
```

- (3) Χρησιμοποιείτε παρενθέσεις για να καθορίζετε ακριβώς και σαφώς τις προτεραιότητες των πράξεων
  - `while ((c=getchar()) != EOF)` → οι παρενθέσεις χρειάζονται, γιατί != >προτ. από =
  - `if (x & MASK == BITS)` είναι ισοδύναμο με
    `if (x & (MASK == BITTS))` αλλά το σωστό είναι μάλλον το
    `if( (x & MASK) == BITS)`
  - Χρησιμοποιείτε () ακόμη και όταν δεν είναι απαραίτητες

```
disekto_etos = y % 4 == 0 && y % 100 != 0 || y % 400 != 400;
vs.
disekto_etos = ((y % 4 == 0) && (y % 100 != 0)) && (y % 400 != 400);
```

- (4) Χωρίστε πολύπλοκες εκφράσεις σε πιο απλές

```
*x = += (*xp = (2*k < (n-m) ? c[k+1] : d[k--]));
vs.
if ( (2*k) < (n-m) )
     *xp = c[k+1];
else
     *xp = d[k--];
*x += *xp;
```

- (5) Χρησιμοποιείτε την γλώσσα για να υπολογίζετε μεγέθη

```
count += 4;
vs.
count += sizeof(int);
```

(Γ) Εκφράσεις

- (6) Γράψτε "καθαρό" κώδικα

```
subkey = subkey >> (bitoff – ((bitoff>>3)<<3));
vs.
subkey = subkey >> (bitoff – (bitoff & ~0x7);
vs.
subkey = subkey >> (bitoff & 0x7);
vs.
subkey >>= (bitoff & 0x7);
```

- (7) Προσοχή σε εκφράσεις με side-effects

  `i = x++;` ➔ well defined

  `str[i++] = str[i++] = ' ';` ➔ ?
  - Πρόθεση: βάλε ένα κενό ' ' στους δύο επόμενους chars
  - Ωστόσο η C δεν καθορίζει την σειρά με την οποία υπολογίζονται οι υποεκφράσεις σε μια έκφραση
  - Αν το i γίνεται update στο τέλος της έκφρασης τότε θα κάνει skip ένα χαρακτήρα:

    str[i] = ' '; str[i] = ' '; i++; i++;

- (8) Μην χρησιμοποιείτε macros αντί για συναρτήσεις και αν χρησιμοποιήσετε macros, χρησιμοποιείτε παρενθέσεις
- (9) Χρησιμοποιείτε "ιδιωματικές" εκφράσεις
  - Κάθε γλώσσα έχει τις δικές της. Η χρήση τους βοηθά να καταλάβουν οι άλλοι τον κώδικά μας πιο γρήγορα.

```
i=0;                    for (i=0; i<n; ) for(i=0; --i>=0; )        for(i=0;i<n;i++)
while (i <= (n-1))        A[i++]=1;          A[i]=1;                  A[i]=1;
  A[i++]=1;
```

  - Όλα είναι σωστά, αλλά στη C ο ιδιωματικός τρόπος να εκφράσουμε ένα loop είναι ο τελευταίος

(Δ) Σχόλια

- (1) Μην αναλύετε το προφανές

```
default: /* default */
break;

/* return SUCCESS */
return SUCCESS;
```

```
/* initialize total to number_received */
 node->total = node->numer_received;

if (c==EOF) { /* if at and of file */
zeroCount++; /* increment zeroCount by one */
```

  - Αφαιρέστε όλα αυτά τα σχόλια
- (2) Βάζουμε σχόλια σε τρία κυρίως σημεία
  - (i) Σε global μεταβλητές και κεντρικά structures του προγράμματος που έχουν σημαντικό ρόλο στο πρόγραμμα συνολικά

```
struct State_s {                        /* prefix + suffix set */
  char *prefix[NPREF];                  /* prefix words */
  Suffix_T *suffix;                     /* list of suffixes */
  struct State_s *next;                 /* next in hash table */
}
```

  - (ii, iii) Σε συναρτήσεις, εκεί που δηλώνονται (interface) και εκεί που υλοποιούνται (.c)
    - Και στις δύο περιπτώσεις βάζουμε το ίδιο ακριβώς σχόλιο
    - Αυτό χρειάζεται ώστε όποιος καλεί την συνάρτηση και όποιος την υλοποιεί να έχουν την ίδια κατανόηση για το τι κάνει η συνάρτηση

```
/* return an integer in the range [0…r-1] */
int random(int r){
          return (int)floor(random()*r);
}
```

Στο σχόλιο, πείτε:

- Τι κάνει η συνάρτηση (όχι πως το κάνει). Χρησιμοποιείστε στην περιγραφή ονόματα παραμέτρων και μεταβλητών.
- Τι επιστρέφει
- Αν διαβάζει/γράφει κάτι από input/output
- Αναφέρετε οτιδήποτε άλλα side-effects έχει (και το I/O είναι μια μορφή side-effect)

(Δ) Σχόλια

- (3) Μην γράφετε σχόλια για ”κακό” κώδικα. Ξαναγράψτε τον κώδικα.

```
if ((person_age < 27) && (person_age >= 15) && (person_age != 19))
vs.
if (isProperAge(person_age))
...
```

- (4) Μην γράφετε σχόλια που είναι αντίθετα στον κώδικα

```
time(&now);
strcpy(date, ctime(&now));
/* get rid of trailing newline after char copies from ctime */
i=0;
while (date[i] >= ' ') i++;
date[i]=0;
```

- Ο κώδικας και το σχόλιο είναι σωστά, αλλά μπερδεύουν. Τυχαίνει το ‘\n’ να είναι ο μόνος printable χαρακτήρας πριν το κενό ‘ ‘. Οπότε το σχόλιο εκφράσει σωστά την πρόθεση και ο κώδικας κάνει το σωστό, αλλά με ”cryptic” τρόπο. Καλύτερα:

```
time(&now);
strcpy(date, ctime(&now));
/* get rid of trailing newline after char copies from ctime */
for (i=0; date[i] != '\n'; i++)
     ;
date[i] = '\0';
or
date[strlen(date)-1]='\0';
```

Reading

- (online) C Style and Coding Standards, Glenn Skinner, Suryakanta Shah, and Bill Shannon. AT&T Information System, Sun Microsystems. [pdf]
- Πολλά από τα παραδείγματα και τους σχετικούς κανόνες προέρχονται από το
  - Chapter 1 of The Practice of Programming, by Brian W. Kernighan and Rob Pike. 2nd Edition, Addison-Wesley, Inc., 1999
- Αν ενδιαφέρεστε περισσότερο, μπορείτε να κοιτάξετε και το
  - The Art of Readable Code. Dustin Boswell, Trevor Foucher. Released November 2011. Publisher(s): O'Reilly Media, Inc. ISBN: 9780596802295