Carta da regalo con LaTeX. Una proposta di gadget di benvenuto per il GUIT

Gianluca Pignalberi

Sommario

Diverse soluzioni programmative per carta da regalo similcommerciale con LaTeX.

Abstract

Several programming solutions for off-the-shelf-like gift paper with LaTeX.

Introduzione

In occasione del GUIT*meeting2019* abbiamo festeggiato gli 80 anni di Claudio Beccari.[1] Il Consiglio Direttivo, in rappresentanza del GUIT, gli ha fatto un piccolo dono.

Che regalo si può fare a un importante ex professore del Politecnico di Torino che ha speso oltre 30 anni al "servizio di TeX" (gli appassionati di fumetti avranno forse sentito qualcosa di familiare: Plazzi e Rosati (1996))? Il comitato ristretto[2] aveva proposto al Direttivo di regalare a Claudio un disegno di Duane Bibby: proposta accettata. Il disegno avrebbe dovuto ritrarre Claudio sotto la Mole Antonelliana attorniato da alcuni membri storici del GUIT intenti a festeggiarlo. Bibby, contattato da Maurizio Himmelmann (uno dei fondatori del GUIT), ha rifiutato il lavoro essendo ormai in pensione.

In quei giorni veniva lanciata su Kickstarter una raccolta fondi per rimettere in sesto una Linotype ubicata a Vienna (Kaps, 2018). Tra le ricompense per i *backer* c'era un set di biglietti da visita con testo di lunghezza massima prefissata e scelto da ogni sottoscrittore. Tale ricompensa sarebbe stata corredata delle tre righe di caratteri (Helvetica) di piombo fuse dalla Linotype ripristinata e usate per la stampa dei biglietti. Il nome dei sottoscrittori sarebbe stato poi "tatuato" sulla macchina, quindi il GUIT avrebbe avuto il suo riconoscimento tangibile per il contributo dato al progetto.

1. In realtà, il suo ottantesimo compleanno cade nel 2020, ma abbiamo preferito approfittare del suo ultimo meeting da direttore di ArsTeXnica nel "suo" Politecnico.

2. Quella dei comitati ristretti è un'idea di Luigi Scarso: tre o quattro membri del Direttivo delegati per uno specifico compito, magari supportati da un volontario non facente parte del Direttivo, tengono riunioni informali tra loro anche senza preavviso per mettere a punto la strategia per portare a termine il compito. La cosa ha funzionato per il meeting e per la domanda all'Anvur.

Figura 1: Il regalo di Claudio Beccari.

Ricevuto per tempo[3] il regalo (visibile nella figura 1), rimaneva il problema di presentarlo: la scatola era un'anonima scatola di cartone pressato grigio, realizzata alla bell'e meglio, coi punti metallici a tenere insieme gli angoli della scatola e del coperchio.

Tale dono, consegnato da Enrico Gregorio e gradito dal festeggiato (Beccari, 2019), andava incartato in una maniera... originale. Ci sarebbe voluta della bella carta da regalo. Ma perché comprarla, quando si poteva realizzarne di personalizzata con LaTeX? Avrebbe dovuto ricordare le carte commerciali con i disegni disposti a scacchiera, ma avere il logo del GUIT al posto di anonimi disegnini. Questa sì!, che sarebbe stata una carta originale.

All'insaputa di tutti decisi di verificare l'esistenza di un servizio di *print-on-demand* (POD d'ora in poi) che realizzasse tale prodotto. Trovatolo, stesi un documento LaTeX per comporre il foglio in maniera conforme alle specifiche richieste dal POD stesso. Le caratteristiche autoimposte per la carta erano le seguenti: sfondo colorato uniformemente; pattern composto dal logo GUIT disposto a scacchiera, variamente dimensionato (da `\large` a `\Huge`) e ruotato di multipli di 45°. Naturalmente, dimensioni e rotazioni sarebbero state casuali.

L'articolo descrive il problema affrontato, cioè la realizzazione della carta da regalo personalizzata GUIT, partendo dall'analisi delle specifiche richieste dal servizio POD (sezione 1). Quindi mostra e commenta le prove preliminari in LaTeX (sezione 2) e il successivo completamento in C delle parti

3. I tempi previsti per il ripristino della Linotype, la realizzazione e la consegna delle ricompense erano pochi mesi: entro gennaio 2019; il regalo è arrivato però alla fine di luglio 2019 a causa di alcuni intoppi legati proprio al ripristino della Linotype.

casuali (sezione 3). Passa poi a descrivere il passaggio finale per l'invio in stampa (sezione 4). Di seguito fornisce una traduzione in LuaLATEX (sezione 5), una in LATEX (sezione 6) e una in LATEX3 (sezione 7) del codice C + LATEX. Dopodiché generalizza il progetto originale della sezione 3, prima adottando un'immagine al posto del logo testuale (sezione 8), poi (sezione 9) rendendo parametriche le dimensioni della tabella (numero di righe e colonne) per adeguare la griglia alle esigenze dell'utente e a quelle della pagina, col conseguente ricalcolo delle dimensioni delle celle. Nelle conclusioni (sezione 10) lancia al nuovo Direttivo una proposta. L'appendice A, destinata solo ai programmatori e agli interessati, parla del sostanziale fallimento dei tentativi di migliorare pur di poco le prestazioni del codice C proposto alla sezione 3. Il suo contenuto prevede solo un minimo di conoscenze sull'aritmetica dei puntatori nel C. Infine, l'appendice B esegue un confronto orientativo dei programmi descritti nelle sezioni 3–7.

1 Il servizio POD

Prima di iniziare questo progetto, è stata fondamentale la ricerca di un servizio POD in grado di realizzare il prodotto voluto. Una rapida ricerca sul web ha fornito il nome di *print24* (PRINT24, 2019), servizio in grado di produrre da uno a n fogli di carta da regalo personalizzati, in diversi pesi e formati.

Nel modulo d'ordine è possibile stabilire il formato del foglio (A0–A3, B1 e B2), il suo orientamento, il peso della carta (115, 135 o 170 g/m^2) e la sua proprietà (opaca o lucida), il tipo di stampa (nero, nero più un Pantone, nero più oro o argento, CMYK o CMYK più uno degli speciali appena nominati) e altri dati necessarî a determinare il costo finale.

Nel caso in esame, la scelta è stata: foglio A3 a singola facciata, orientato orizzontalmente e carta da 115 g/m^2 opaca.

2 Il documento preliminare

Il sorgente LATEX per la produzione di un foglio con un pattern posizionato a griglia è in sé piuttosto banale: si tratta di riempire una tabella adeguatamente dimensionata alternando le celle vuote e piene in base alle righe e alle colonne. Sappiamo da Beccari *et al.* (2016) che il problema può essere risolto con minori o maggiori "compattezza" ed "espressività" computazionale in base al linguaggio scelto. Nel caso esaminato in questo articolo, la tabella sarà una vera tabella "di testo" e non un disegno e il pattern di riempimento sarà un logo di testo anziché una campitura di colore.

Il primo passo è stato fatto scrivendo una tabella 3×3, logo non ruotato e monodimensionale il cui risultato è visibile nella figura 2:

```
1  \documentclass[12pt]{standalone}
2
3  \usepackage[T1]{fontenc}
4  \usepackage[utf8]{inputenc}
5  \usepackage{guit}
6
7  \begin{document}
8  \noindent\begin{tabular}{@{}ccc@{}}
9  \GuIT* & & \GuIT* \\
10 & \GuIT* & \\
11 \GuIT* & & \GuIT* \\
12 \end{tabular}
13 \end{document}
```

Figura 2: Risultato della compilazione della versione preliminare della carta da regalo.

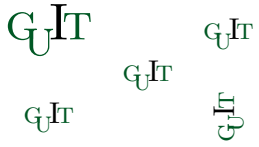

Figura 3: Risultato della compilazione della versione preliminare della carta da regalo con ridimensionamento e rotazione del logo.

Prima di espandere la tabella alle dimensioni finali, la sperimentazione è andata avanti per verificare gli altri vincoli progettuali: la rotazione e il ridimensionamento del logo. Ecco il nuovo contenuto della tabella coi nuovi numeri di riga dopo l'inclusione di graphicx per avere il comando `\rotatebox`:

```
9  \LARGE\GuIT* & & \GuIT* \\
10 & \GuIT* & \\
11 \GuIT* & & \rotatebox[origin=c]{90}{\
       GuIT*} \\
```

Il nuovo risultato è visibile nella figura 3

Fin qui non c'è niente di particolarmente difficile: è anzi tutto banale. Riempire una tabella dalle dimensioni definitive di 20×16,[4] riempiendone le celle dispari in una riga e quelle pari nell'altra riga è solo un po' più lungo: ovviamente basta scrivere per esteso le prime due righe e copiarne il contenuto nelle altre. La vera sfida è fare in modo che ogni logo sia dimensionato e ruotato casualmente, sebbene con dimensioni e angoli prefissati: farlo a mano per una tabella di nove caselle è accettabile; per una tabella di 320 caselle lo è meno, volendo mantenere la casualità. Senza contare che lo stesso metodo potrebbe essere applicato alla produzione di carta da parati e il numero delle celle della ta-

4. Queste dimensioni sono state decise pensando a fogli di formato A4 e A3 e sono state mantenute fisse per tutto il progetto. Naturalmente, nessuno ne impone la fissità e vedremo come stabilirle arbitrariamente nella sezione 9.

bella potrebbe essere ancora maggiore rispetto a quante ne può contenere un già grande foglio A0.

Un'altra cosa di cui tener conto è la dimensione verticale delle celle, che non è possibile demandare all'intestazione della tabella.

3 La versione definitiva: C + LaTeX

Il linguaggio C è stato il mio strumento di lavoro dal quinto anno di scuola superiore fino a qualche anno dopo il termine della mia ormai lontana carriera di studente universitario. Nonostante la ruggine, il ricorso al C è stata la scelta più logica per risolvere la parte mancante in breve tempo (neanche due settimane tra il momento della nascita dell'idea della carta e la data di consegna del regalo).

Il progetto completo consta di due file preliminari (un generatore di dati in C, `makeGPtv1.c`, e un utilizzatore di dati in LaTeX, `giftpaper.tex`), e del file di dati (`table.tex`) atto a "collegare" i due preesistenti. Verrà dunque applicata la tecnica descritta esaustivamente in Giacomelli e Pignalberi (2015).

Il primo file preliminare, che è anche l'unico a essere eseguito, contiene questo codice:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <time.h>
5
6  #define DEGSTEP 45
7  #define MAXROTSTEP 8
8  #define NUMDIM 5
9  #define NROWS 16
10 #define NCOLS 20
11 #define MAXLEN 255
12
13 FILE *open_file (char *, char *);
14
15 int main ()
16 {
17   short i, j, rotation, dimpos;
18   char *dimen[NUMDIM] = {"\\large", "
         \\Large", "\\LARGE", "\\huge", "
         \\Huge"},
19        tabular[NROWS][NCOLS][MAXLEN],
              fg[] = "table.tex", tmp[
              MAXLEN];
20   FILE *table;
21
22   srand (time (NULL));
23   for (i = 0; i < NROWS; i++)
24     for (j = 0; j < NCOLS - 1; j++)
25       strcpy (tabular[i][j], "␣&");
26   for (i = 0; i < NROWS; i++)
27     strcpy (tabular[i][NCOLS - 1], "␣
           \\\\");
28   for (i = 0; i < NROWS; i++)
29     for (j = i % 2; j < NCOLS; j += 2)
           {
30       rotation = ((int) rand() %
             MAXROTSTEP ) * DEGSTEP;
31       dimpos = (int) rand() % NUMDIM;
32       sprintf (tmp, "\\begin{minipage
             }{2.1cm}\\vbox␣to␣1.856cm{\\
             vfill\\hfil\\rotatebox[
             origin=c]{%d}{%s\\GuIT*}\\
             hfill\\vfill}\\end{minipage}
             ␣%s", rotation, dimen[dimpos
             ], tabular[i][j]);
33       strcpy (tabular[i][j], tmp);
34     }
35   if ((table = open_file (fg, "w")) ==
        NULL) exit (1);
36   for (i = 0; i < NROWS; i++)
37     for (j = 0; j < NCOLS; j++)
38       fprintf (table, "%s", tabular[i
             ][j]);
39   fclose (table);
40   system ("pdflatex␣giftpaper");
41   return 0;
42 }
43
44 FILE * open_file (char *nomefile, char
       *accesso)
45 {
46   FILE *fp;
47
48   if ((fp = fopen (nomefile, accesso))
        == NULL)
49     fprintf (stderr, "Errore␣nell'
           apertura␣di␣%s\n", nomefile);
50   return fp;
51 }
```

Il programma si apre col caricamento di alcune librerie standard e con la definizione di alcune costanti (righe 1–11). La funzione `open_file`, dichiarata all'inizio (riga 13) e definita alla fine (righe 44–51), serve solo ad aprire un file comunicandone l'esito. La funzione `main`, dopo aver dichiarato e/o definito alcune variabili (tra cui gli array[5] contenenti la matrice che replica in memoria l'intera tabella dei pattern e il vettore delle possibili dimensioni del logo), svolge il grosso del lavoro. Dapprima inizializza il generatore di numeri pseudocasuali (riga 22), quindi inizializza la matrice: i primi due cicli (righe 23–25 e 26–27) inseriscono rispettivamente i separatori di cella (`&`) nelle celle delle prime $n-1$ colonne e i separatori di riga (`\\`) nelle celle dell'ultima colonna. Il ciclo alle righe 28–34 riempie la matrice con i dati effettivi: il primo `for` scandisce tutte le righe dell'array

5. Non bisogna dimenticare che in C gli array iniziano con la cella nº 0. Normalmente vengono chiamati "vettori" se monodimensionali e "matrici" se pluridimensionali, ma è chiaro che un vettore di stringhe è, di fatto, una matrice di caratteri.

(cioè si muove in verticale dall'alto verso il basso); il secondo scandisce le colonne (cioè si muove orizzontalmente) saltando una posizione ogni due e partendo da 0 o da 1 in base al valore della riga (riga 29 del codice; l'Appendice approfondirà proprio questo ciclo); nella cella identificata a ogni passo del ciclo verrà scritto (riga 33) il codice LaTeX che dimensiona correttamente la cella e ci pone il logo con l'angolo di rotazione e la dimensione generati alle righe 30 e 31[6] completato col terminatore precedentemente scritto nella cella (riga 32). Le righe 35–39 scaricano il contenuto della matrice in memoria nel file `table.tex` e la riga 40 demanda al sistema operativo la compilazione del master `.tex`, il cui codice è riportato più avanti. Infine, la riga 41 termina l'esecuzione dando un codice 0, valore standard per segnalare un'esecuzione andata a buon fine.

Essendo questo un file C, andrà compilato seguendo le istruzioni del proprio compilatore. La sua esecuzione genera come sottoprodotto il file contenente la tabella da porre sul foglio e di cui mostriamo un esempio delle prime due celle delle prime due righe:

```
1 \begin{minipage}{2.1cm}\vbox to 1.856
      cm{\vfill\hfil\rotatebox[origin=c
      ]{0}{\Huge\GuIT*}\hfill\vfill}\end
      {minipage} & &
2 & \begin{minipage}{2.1cm}\vbox to
      1.856cm{\vfill\hfil\rotatebox[
      origin=c]{315}{\large\GuIT*}\hfill
      \vfill}\end{minipage} &
```

Quest'ultimo è il file "di raccordo" e viene incluso in `giftpaper.tex`, così da avere la tabella perfettamente composta. Come si può notare, le dimensioni orizzontali delle celle sono esattamente un ventesimo del lato maggiore di un foglio A3, mentre le dimensioni verticali sono un sedicesimo del lato minore dello stesso foglio. Le dimensioni sono precalcolate perché il progetto non è nato per essere modulare. Nella sezione 9 vedremo come rendere più versatile il prodotto.

Il contenuto di `giftpaper.tex` è il seguente:

```
1 \documentclass[12pt]{article}
2
3 \usepackage[T1]{fontenc}
4 \usepackage[utf8]{inputenc}
5 \usepackage{guit}
6 \usepackage{tabu}
7 \usepackage[a3paper,hmargin=0pt,
      vmargin=0pt,landscape]{geometry}
8 \usepackage[center,width=42.4cm,height
      =30.1cm]{crop}
9 \usepackage{xcolor}
10 \usepackage{graphicx}
11
12 \begin{document}
13 \pagecolor{black!5!yellow!30}
14 \noindent\begin{tabu} to \textwidth {@
       {}*{20}X}
15 \input{table}
16 \end{tabu}
17 \end{document}
```

A tabular, usato nella sezione precedente, è stato preferito tabu per la possibilità di specificare la larghezza della tabella e di equidimensionare le singole celle. Non sono stati usati gli specificatori di posizione orizzontale e verticale dell'ambiente tabu: per la presenza di minipage e `\vbox` nel codice da scrivere nelle celle sono stati usati direttamente comandi di TeX. Questo rende l'uso di tabu forse eccessivo e gli si potrebbe preferire tabularx.

Sicuramente si nota il fatto che la tabella inizia esattamente sul bordo della gabbia di pagina (cioè sul bordo sinistro del foglio; lo spessore naturale delle celle di una tabella si elimina scrivendo `@{}` sul lato della cella da cui lo si vuole eliminare) ma non termina sul bordo destro. Questo è voluto perché i logo alle dimensioni massime finiscono troppo vicino al margine destro, rovinando la simmetria della tabella rispetto al foglio.

Il risultato dell'esecuzione del programma in C è un PDF col colore di sfondo scelto e una serie di logo GuIT (a colori) dimensionati e ruotati casualmente posti a scacchiera su una griglia regolare posta su un foglio di dimensione totale A3 più le abbondanze richieste dal POD.

4 Postproduzione per la stampa

Dopo aver prodotto il PDF definitivo, che il POD vuole senza crocini di taglio o altri segni e visibile nella figura 4, rimane da applicare la proprietà relativa al modello di colore. La richiesta del POD è un file PDF, JPEG o TIFF a 300 dpi minimo e nella modalità colore CMYK con profilo colore Fogra39L per la stampa su carta patinata o Fogra47L per la stampa su carta riciclata o usomano.

Non avendo a disposizione un programma commerciale tipo InDesign, ho importato il PDF di LaTeX in un documento Scribus e ho esportato quest'ultimo come PDF dopo aver selezionato la modalità stampa (contrapposta alle modalità schermo/web e toni di grigio) e impostato l'unico profilo colore disponibile nell'installazione standard (Fogra27L, dato per uso con carta patinata). Il file è stato accettato senza alcun problema dal POD e il risultato è decisamente conforme alle aspettative o comunque abbastanza buono.

5 La traduzione in LuaLaTeX

Gli utenti di LuaTeX (LuaTeX development team, 2019) troveranno oggettivamente più sem-

6. Avremmo potuto risparmiare qualche byte di memoria e qualche millisecondo di tempo di accesso scrivendo il codice che genera i due valori direttamente nella `sprintf`, ma ciò a scapito della leggibilità del codice.

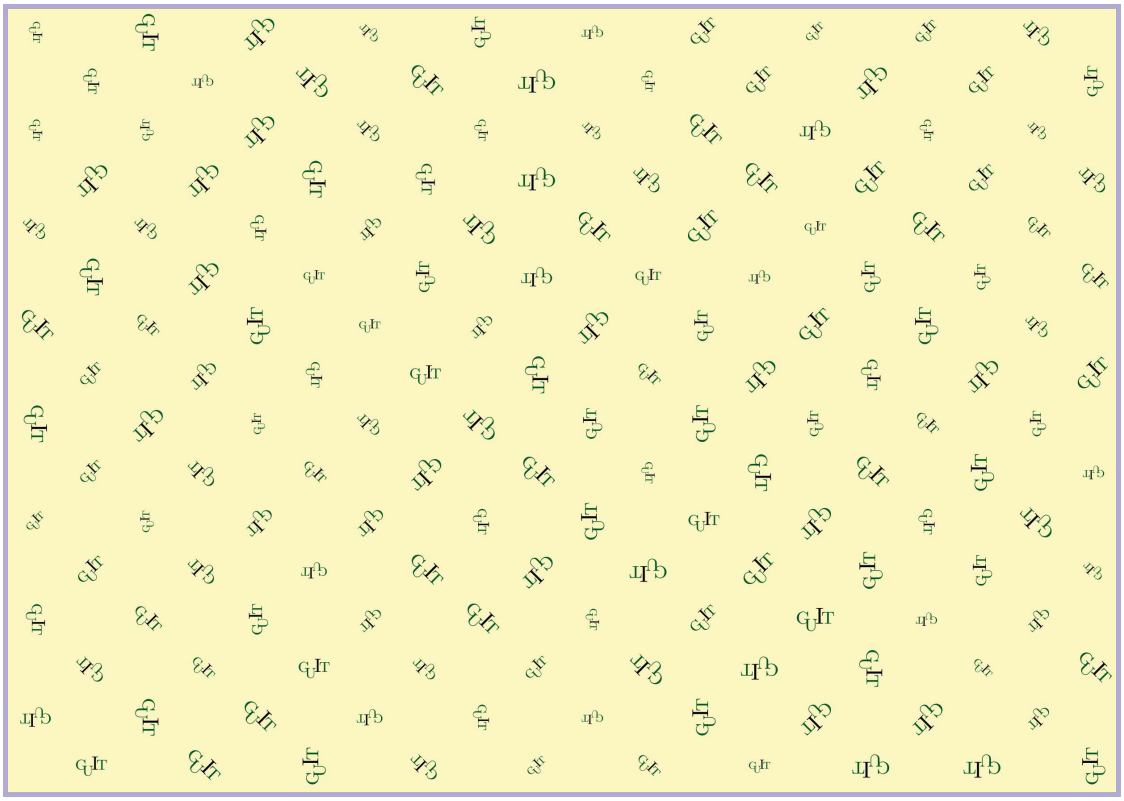

FIGURA 4: Il foglio definitivo mandato in stampa. Le abbondanze sono evidenziate artificialmente solo a beneficio dei lettori.

plice risolvere il problema grazie a Lua. Sebbene io abbia sentito parlare di Lua per la prima volta nel 2001 durante il mio periodo come borsista al CASPUR (ora parte del CINECA), non ho mai studiato né utilizzato tale linguaggio. Dunque, questa sezione descrive la mia prima esperienza d'uso di Lua puro e applicato.

Come già visto nella sezione 3, dobbiamo rendere pseudocasuale l'angolo di rotazione e la dimensione del logo di testo. Ci serviranno dunque due funzioni: una che generi un numero casuale $N \in \{0, 45, 90, 135, 180, 225, 270, 315\}$ e una che restituisca una dimensione compresa tra `\large` e `\Huge`. Come ampiamente discusso nell'implementazione originale in C, quest'ultima funzione si realizza generando un numero casuale che fungerà da indice dell'array in cui abbiamo memorizzato le stringhe coi comandi delle dimensioni volute. Il resto è banale applicazione delle chiamate a Lua. Il codice per ottenere la carta da regalo è il seguente:

```
\documentclass[12pt]{article}

\usepackage{fontspec}
\usepackage{guit}
\usepackage{tabu}
\usepackage[a3paper,hmargin=0pt,
    vmargin=0pt,landscape]{geometry}
\usepackage[center,width=42.4cm,height
    =30.1cm]{crop}
\usepackage{xcolor}
\usepackage{graphicx}
\usepackage{luacode}

\begin{luacode}
local dimen = {"\\large", "\\Large",
    "\\LARGE", "\\huge", "\\Huge"}
local Trows = 16
local Tcols = 20
local anglestep = 45
local mt = {}
function randrot()
  tex.print(math.random(0,7) *
      anglestep)
end
function randimen()
  tex.print(dimen[math.random(5)])
end
function maketable()
  for i = 0, Trows - 1 do
    mt[i] = {}
    for j = 0, Tcols - 2 do
      mt[i][j] = " &"
    end
  end
  for i = 0, Trows - 1 do
    mt[i][Tcols - 1] = " \\\\ "
  end
  for i = 0, Trows - 1 do
```

```
35      for j = i % 2, Tcols - 1, 2 do
36        mt[i][j] = "\\cell" .. mt[i][j]
37      end
38    end
39    tab = io.open ("table.tex", "w")
40    for i = 0, Trows - 1 do
41      for j = 0, Tcols - 1 do
42        tab:write (mt[i][j])
43      end
44    end
45    tab:close ()
46  end
47  \end{luacode}
48
49  \newcommand\cell{\begin{minipage}{2.1
        cm}\vbox to 1.856cm{\vfill\hfil\
        rotatebox[origin=c]{\directlua{
        randrot()}}{\directlua{randimen()}
         \GuIT*}\hfill\vfill}\end{minipage
        }}
50  \begin{document}
51  \pagecolor{black!5!yellow!30}
52  \directlua{maketable()}
53  \noindent\begin{tabu} to \textwidth {@
        {}*{20}X}
54  \input{table}
55  \end{tabu}
56  \end{document}
```

La riga 10 include il pacchetto per avere un ambiente che faciliti la scrittura del codice Lua; le righe 12 e 47 aprono e chiudono questo ambiente. Le righe 13–23 contengono il codice Lua discusso in precedenza: la funzione `randrot` fornisce un angolo di rotazione compreso tra 0° e 315° a intervalli di 45°; la funzione `randimen` restituisce una delle dimensioni comprese nella tabella[7] `dimen`. Infine, le righe 24–46 definiscono una funzione che, al pari di quanto visto nella sezione 3, inizializza una tabella Lua con `&` in tutte le colonne meno l'ultima (righe 25–30), dove scrive `\\` (righe 31–33), quindi antepone a questi caratteri il comando `\cell` solo dove occorre (celle pari delle righe pari e celle dispari delle righe dispari, righe 34–38); infine scrive l'array in un file esterno che verrà incluso entro l'ambiente `tabu` (righe 39–45).

Per evitare di scrivere un codice tanto lungo quanto poco manutenibile, definiamo il comando `\cell` (riga 49) nel modo affine a quello descritto nella sezione 3. Con LuaLaTeX, però, possiamo fare le chiamate `\directlua` per generare i valori pseudocasuali nella macro di LaTeX e dimenticare il metodo bizantino richiesto dal C, cioè generare una stringa col comando e i valori pseudocasuali da scrivere direttamente nella tabella, senza possibilità di condensare tutto in un comando LaTeX. Quando all'interno della tabella dobbiamo riempire una cella, ci basterà dare il comando `\cell`.

7. Qui il termine tabella si riferisce al tipo di dato Lua e non a un ambiente `tabular` di LaTeX.

6 Non conosci il pacchetto lcg?

Arrivati a questo punto, molti degli utenti principianti e senza esperienza di programmazione potrebbero obiettare che le soluzioni offerte finora sono troppo esoteriche e potrebbero volere una soluzione in LaTeX puro. Un Egregio (o Esperto) Guru potrebbe allora indicare la via dicendo: «Perché vuoi uccidere le mosche col C(annone) o col raggio l(u)aser quando puoi usare un comodo schiacciamosche? Non conosci il pacchetto lcg?» e dimostrando così la sua conoscenza enciclopedica dell'ecosistema di TeX e la mia miopia non solo ottica (e poi mi rimanderebbe pure a Stack Exchange (2015) per farmi valutare una soluzione basata su Ti*k*Z).

Effettivamente, il pacchetto in questione contiene delle macro per generare numeri pseudocasuali che verranno memorizzati in un contatore definibile dall'utente. Anche a beneficio dei non programmatori C o Lua, diamo una soluzione al problema basata su lcg.

Il codice, che trovate in `giftpaperlatex.tex` nei sorgenti di accompagnamento, è il seguente:

```
1  \documentclass[12pt]{article}
2
3  \usepackage[T1]{fontenc}
4  \usepackage{guit}
5  \usepackage{tabu}
6  \usepackage[a3paper,hmargin=0pt,
       vmargin=0pt,landscape]{geometry}
7  \usepackage[center,width=42.4cm,height
       =30.1cm]{crop}
8  \usepackage{xcolor}
9  \usepackage{graphicx}
10 \usepackage{lcg}
11
12 \makeatletter
13 \def\nloop#1{%
14   \def\nl@@p##1##2\repeat#1{%
15   \def##1{##2\relax\expandafter##1\fi}
         %
16    ##1\let##1\relax}%
17 \expandafter\nl@@p\csname nl@@p-\
       string#1\endcsname
18 }
19 \makeatother
20 \newcommand\dimension[1]{%
21   \ifcase #1\relax
22   \or \large
23   \or \Large
24   \or \LARGE
25   \or \huge
26   \or \Huge
27   \fi}
28 \newcounter{x}
29 \newcommand\cell{\begin{minipage}{2.1
       cm}\vbox to 1.856cm{\vfill\hfil\
       chgrand [first=0, last=7]\setbox
       1=\hbox{\rand}\hskip-\wd1\multiply
       \value{rand} by 45 \value{x}=\
```

```
      value{rand}\rotatebox[origin=c]{\
      value {x}}{\chgrand[first=1, last
      =5]\setbox1=\hbox{\rand}\hskip-\wd
      1\dimension{\value{rand}}\GuIT*}\
      hfill\vfill} \end{minipage}}
30
31  \begin{document}
32  \newwrite\tempfile
33  \immediate\openout\tempfile=table.tex
34  \pagecolor{black!5!yellow!30}
35  \newcounter{Trows}\setcounter{Trows
        }{16}
36  \newcounter{Tcols}\setcounter{Tcols
        }{20}
37  \newcounter{nrows}
38  \newcounter{ncols}
39  \value{nrows}=0
40  \nloop\a
41    \value{ncols}=0
42    \nloop\b
43      \ifodd\value{nrows}%
44        \ifodd\value{ncols}\immediate\
              write\tempfile{\unexpanded{\
              cell}}\fi
45      \else%
46        \ifodd\value{ncols}\relax\else\
              immediate\write \tempfile{\
              unexpanded{\cell}}\fi
47      \fi
48      \stepcounter{ncols}
49      \ifnum\value{ncols}<\value {Tcols
            }\immediate\write \tempfile{ &}
            %
50    \repeat\b
51      \immediate\write\tempfile{\
            unexpanded{\\}}\stepcounter{
            nrows}
52    \ifnum\value{nrows}<\value{Trows}%
53  \repeat\a
54  \immediate\closeout\tempfile
55  \noindent\begin{tabu} to \textwidth {@
        {}*{20}X}
56  \input{table}
57  \end{tabu}
58  \end{document}
```

Al pari delle versioni precedenti, il documento mette insieme le celle della tabella in un file (`table.tex`, soluzione proposta da Reza Afzalan su https://tex.stackexchange.com/questions/50296/problem-with-using-loop-inside-the-tabular-environment) che verrà poi incluso nell'ambiente tabu per la composizione. Vediamo nel dettaglio come fa. Alla riga 10 include il pacchetto lcg. Alle righe 12–18 definisce una versione di `\loop` (proposta da David Carlisle su https://tex.stackexchange.com/questions/58049/nested-loop-macro) che permette gli annidamenti. Alle righe 20–27 fornisce una macro (`\dimension`) che, in base al valore numerico ricevuto in input, emette un comando di dimensionamento il cui "valore" è compreso tra `\large` e `\Huge` (simuliamo in questo modo quanto abbiamo finora fatto con gli array o con le tabelle). Alla riga 28 viene creato un contatore (`x`), usato nel successivo comando `\cell` (scritto alla riga 29). Quest'ultimo, al netto di quanto visto anche nelle occasioni precedenti (dimensionamento orizzontale e verticale della cella, centratura della scritta) fa le seguenti cose: imposta la generazione di un numero compreso tra 0 e 7 (comando `\chgrand`) e lo genera (comando `\rand`). Questo numero, moltiplicato per 45 e messo nel contatore `x`,[8] viene usato come angolo per la `\rotatebox`. Dopodiché viene impostata la generazione di un numero compreso tra 1 e 5 che sarà l'argomento del comando `\dimension`. Alle righe 32–33 si definisce un file temporaneo (`table.tex`) che viene aperto in scrittura. Le righe 35–36 definiscono due contatori contenenti il numero totale di righe e colonne della tabella, queste ultime scandite grazie ai due contatori creati alle righe 37–38. I cicli annidati lavorano in questo modo: si parte dalla prima riga scandendo tutte le sue celle; se la riga e la colonna sono entrambe pari (ciò si verifica nelle celle (0,0), (0,2), (0,4)…) o entrambe dispari (cioè (1,1), (1,3), (1,5)…), si scrive sul file un comando `\cell` e, se non abbiamo riempito l'ultima colonna, un divisore di colonna `&`; dopo l'ultima colonna si emette `\\` e si prosegue alla nuova riga; se la riga appena scandita è ultima, il ciclo termina e il file `table.tex` viene chiuso.

Qualcuno si chiederà perché il comando `\rand` viene messo in una scatola la cui larghezza è poi usata per spostarsi a sinistra nel testo. Come avrete visto (e avrete letto nel manuale di lcg) l'uso del numero generato è conseguente alla sua generazione. Sebbene nel manuale non sia esplicitato (ma si vede negli esempî), l'uso di `\rand` inserisce uno spazio non richiesto nel testo e ciò implicherebbe lo spostamento a destra delle scritte, prima centrate nelle celle della tabella.

Un'ultima osservazione può essere fatta sull'inefficienza di scrivere il contenuto della singola cella o del divisore nel file. Sarebbe molto più comodo e veloce comporre un'unica stringa col contenuto di una riga e poi scrivere quest'ultima all'interno del file. Inoltre, il modo scelto mette su una riga il contenuto di una singola cella. Questo si riflette in qualche modo sull'efficienza? Ne discuteremo brevemente nell'appendice B. I lettori potranno cimentarsi con la miglioria proposta partendo dal codice fornito con l'articolo e dal suggerimento dato in https://tex.stackexchange.com/questions/74707/how-to-concatenate-strings-into-a-single-command.

8. Se pensate di usare direttamente il contenuto di `rand` come angolo per la `\rotatebox`, vi ritroverete frustrati perché, quando `\value{rand}` viene effettivamente usato, non è stato ancora moltiplicato per 45 nella catena di espansione delle macro.

7 L'alternativa... nativa: LaTeX3

Il Guru della precedente sezione si sarà chiesto finora perché non ho pensato subito a risolvere il problema con LaTeX3. La risposta, banale, è la stessa di LuaTeX: non ne avevo esperienza e non volevo rischiare di sforare i tempi di produzione. Dunque anche questa sezione rappresenta la mia prima esperienza di programmazione in LaTeX3 e senz'altro la soluzione proposta potrà essere migliorata.

Il codice, contenuto nel file `giftpaper3.tex`, è una mera traduzione del precedente LaTeX ed è il seguente:

```
1  \documentclass[12pt]{article}
2
3  \usepackage[T1]{fontenc}
4  \usepackage{guit}
5  \usepackage{tabu}
6  \usepackage[a3paper,hmargin=0pt,
       vmargin=0pt,landscape]{geometry}
7  \usepackage[center,width=42.4cm,height
       =30.1cm]{crop}
8  \usepackage{xcolor}
9  \usepackage{graphicx}
10 \usepackage{expl3}
11
12 \ExplSyntaxOn
13 \cs_new:Npn \dimension #1
14 { \if_case:w #1 \relax
15     \or: \large
16     \or: \Large
17     \or: \LARGE
18     \or: \huge
19     \or: \Huge
20   \fi}
21 \cs_new:Npn \cell
22 {\begin{minipage}{2.1cm}\vbox to 1.856
       cm{\vfill\hfil\rotatebox[origin=c
       ]{\fp_eval:n {45*randint(0,7)}}{\
       dimension{\fp_eval:n{randint(1,5)
       }}\GuIT*}\hfill\vfill}\end{
       minipage}}
23 \ExplSyntaxOff
24
25 \begin{document}
26 \pagecolor{black!5!yellow!30}
27 \ExplSyntaxOn
28 \iow_new:N \tempfile
29 \iow_open:Nn \tempfile {table.tex}
30 \int_new:N\Trows\int_set:Nn\Trows{16}
31 \int_new:N\Tcols\int_set:Nn\Tcols{20}
32 \int_new:N\nrows\int_set:Nn \nrows{0}
33 \int_new:N\ncols
34 \int_do_while:nNnn{\nrows}<{\Trows}{%
35   \int_set:Nn\ncols{0}
36   \int_do_while:nNnn{\ncols}<{\Tcols}{
         %
37     \int_if_odd:nTF{\nrows}{%
38       \int_if_odd:nTF{\ncols}{%
39         \iow_now:Nn \tempfile {\cell}
40       }{\relax}
41     }{%
42       \int_if_odd:nTF\ncols{\relax}{
           %
43         \iow_now:Nn \tempfile {\cell}
44       }%
45     }
46     \int_incr:N\ncols
47     \int_compare:nNnTF{\ncols}<{\Tcols
         }{%
48       \iow_now:Nn\tempfile{ &}
49     }{\relax}
50   }
51   \iow_now:Nn \tempfile {\\}
52   \int_incr:N\nrows
53 }
54 \iow_close:N \tempfile
55 \ExplSyntaxOff
56 \noindent\begin{tabu} to \textwidth {@
       {}*{20}X}
57 \input{table}
58 \end{tabu}
59 \end{document}
```

La riga 10 include il pacchetto che abilita l'uso delle funzioni[9] di LaTeX3 e, dopo averne abilitato la sintassi (riga 12), definiamo le due funzioni `\dimension` (righe 13–20) e `\cell` (righe 21–22). Queste fanno esattamente quel che facevano le omonime funzioni nella versione LaTeX, ma si avvantaggiano delle funzioni di nuova introduzione. In particolare, `\dimension` trae vantaggio da `\if_case:w`, l'equivalente di `\ifcase`, mentre `\cell` usa la funzione `\fp_eval:Npn` e il generatore di numeri pseudocasuali entro un intervallo (`randint`) messi a disposizione dal modulo per i calcoli in virgola mobile di LaTeX3. Alla riga 23 disabilitiamo la sintassi di LaTeX3.

Al pari della versione LaTeX, anche qui ho preferito scrivere la sequenza di operazioni direttamente nel corpo del documento, senza ricorrere a una funzione come nel caso (obbligato) della versione LuaLaTeX. Detta sequenza, come di consueto aperta con `\ExplSyntaxOn` alla riga 27 e chiusa con `\ExplSyntaxOff` alla riga 55, è l'esatta traduzione di quanto visto e spiegato nella sezione 6 e che non ripeterò. Potete comunque fare riferimento a The LaTeX3 Project (2019) per ogni informazione sul significato dei comandi usati, sulla loro interfaccia e sugli esempî d'utilizzo.

8 Solo testo? E le immagini?

Esaurita la digressione sui linguaggi "altri", torniamo al mio caro vecchio C tenendo in mente che tutte le modifiche di questa sezione sono imple-

9. Ricordiamo essere questa la nomenclatura adottata da LaTeX3.

mentabili "gratis" o quasi nei sorgenti in LuaTEX, in LATEX e in LATEX3.

Possiamo pretendere di scrivere un progetto del genere, pur semplice e breve, e costringerlo a usare solo il logo del GUIT? Ovviamente no!, e questa sezione mostrerà come fare a usare un'immagine al posto del logo.

I cambiamenti principali da fare rispetto al sorgente originale sono due: 1) il vettore delle dimensioni non dovrà più contenere dei comandi LATEX ma dei fattori di scala, quindi dei valori *floating point* (per comodità saranno sempre cinque valori, compresi tra 0.75 e 1.25) e non delle stringhe di caratteri; 2) il comando che forma la stringa da mettere in ogni cella non conterrà più il comando del logo ma un `\includegraphics`:

```
18    float dimen[NUMDIM] = {.75, .875, 1,
          1.125, 1.25};

32        sprintf (tmp, "\\begin{minipage
              }{2.1cm}\\vbox␣to␣1.856cm{\\
              vfill\\hfil\\rotatebox[
              origin=c]{%d}{\\
              includegraphics[scale=%f]{
              tux.pdf}}\\hfill\\vfill}\\
              end{minipage}␣%s", rotation,
               dimen[dimpos], tabular[i][j
              ]);
```

L'immagine usata nella figura 5 è il Tux così come disponibile in Scribus.

Il file completo si trova nei sorgenti col nome di `makeGPiv1.c`.

9 Via, verso nuove dimensioni!

Finora abbiamo lavorato con fogli A3, la dimensione minima prevista dal POD di riferimento. È ora di rendere possibile la selezione delle altre dimensioni disponibili. Lasciare immutate le dimensioni della tabella comporterà produrre fogli più grandi troppo vuoti, ma anche troppo pieni se si deciderà di produrne di più piccoli. Dunque sarà opportuno, se non ingrandire o rimpicciolire le dimensioni del testo o del disegno da porre sul foglio, almeno dare la possibilità di variare arbitrariamente il numero di righe e di colonne della tabella. In questa sezione applicheremo questa modifica al file C, che sarà di conseguenza adattato anche per calcolare le dimensioni fisiche delle celle, laddove prima tali dimensioni erano prefissate. Per completezza potremo scegliere di includere del testo o un'immagine, integrando dunque le due tipologie di oggetti visti separatamente nelle sezioni 3 e 8.

Il sorgente, contenuto nel file `makeGP.c` e stavolta commentato direttamente solo nei punti di nuova introduzione, è il seguente:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <time.h>
5  #include <ctype.h>
6
7  #define DEGSTEP 45
8  #define MAXROTSTEP 8
9  #define NUMDIM 5
10 #define NROWS 16
11 #define NCOLS 20
12 #define MAXLEN 255
13
14 FILE *open_file (char *, char *);
15 int max (int, int);
16
17 int main (argc, argv)
18   int argc;
19   char *argv[];
20 {
21 /* argv[1]: flag testo/immagine.
       Valori leciti: T e I
22  * argv[2]: formato del foglio. Valori
       leciti: A0, A1, A2, A3, B1, B2
23  * argv[3]: numero di righe (celle
       verticali)
24  * argv[4]: numero di colonne (celle
       orizzontali)
25  * argv[5]: testo (anche comando) o
       nome immagine */
26 /* Controllo preliminare sul numero di
       argomenti: 5 parametri + il
      comando */
27   if (argc < 6) {
28     printf ("Parametri␣insufficienti.␣
         Uso:\n␣makeGP␣<tipo␣di␣pattern
         >␣(T␣o␣I,␣testo␣o␣immagine)␣<
         formato␣carta>␣(A0,␣A1,␣A2,␣A3,
         ␣B1,␣B2)␣<numero␣celle␣
         verticali>␣<numero␣celle␣
         orizzontali>␣<testo␣(se␣
         comando,␣\\␣dev'essere␣
         raddoppiato;␣es.␣\\\\textcolor
         {red}{A})␣o␣nome␣del␣file␣con␣
         l'immagine>");
29 /* Se gli argomenti inseriti sono
       insufficienti si termina con
      codice -1 */
30     exit (-1);
31   }
32 /* nRows e nCols prendono il numero di
       righe e colonne della tabella
      dalla riga di comando. Se negativi
       o nulli, tali valori sono portati
       a 1 */
33   short i, j, rotation, dimpos, nSheet
        = 9, nRows = max(atoi(argv[3])
       ,1), nCols = max(atoi(argv[4])
       ,1);
34   float cellWidth, cellHeight;
35 /* Solo la prima lettera del pattern
      è significativa */
```

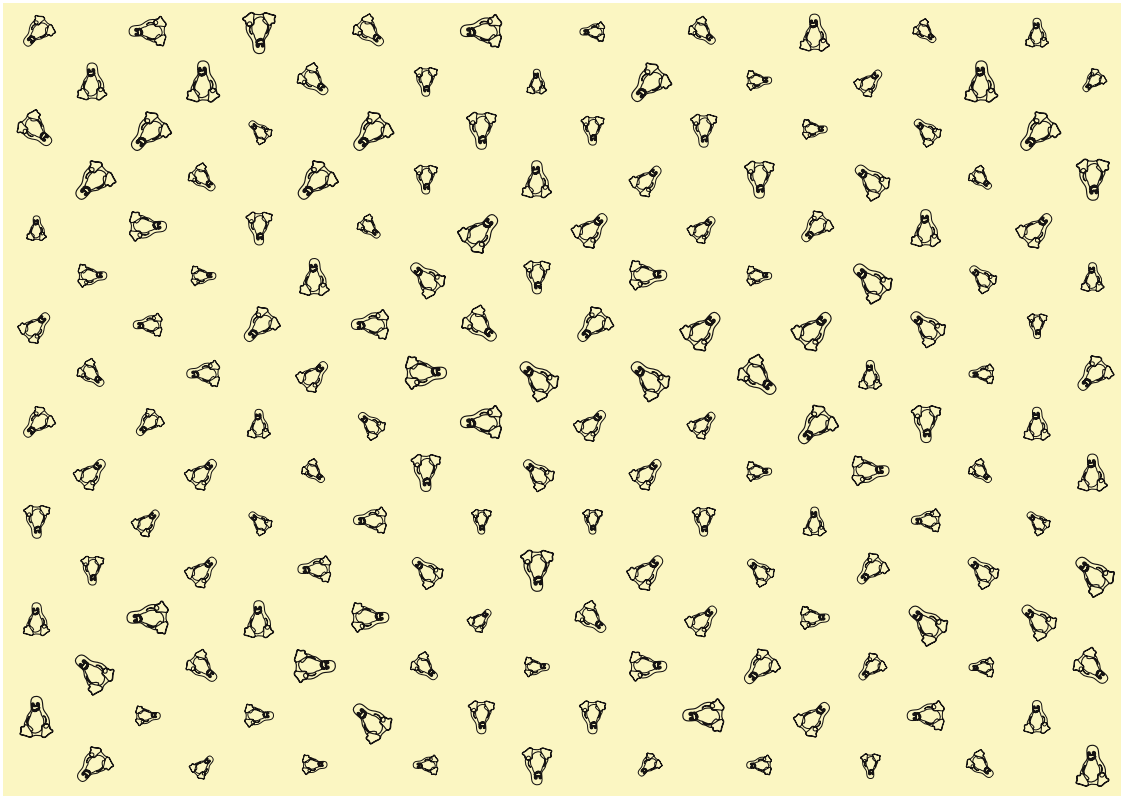

FIGURA 5: Il foglio di carta da regalo con un'immagine al posto di un logo testuale.

```
36    char tabular[nRows][nCols][MAXLEN],
          fg[] = "table.tex", tmp[MAXLEN],
           patternName[MAXLEN/10], pattern
           = toupper(argv[1][0]),
          sheetType[2], realPattern[MAXLEN
          /2];
37    struct sheets {
38      char latexname[MAXLEN/10];
39      float sheetWidth, sheetHeight;
40    };
41  /* Questa variabile contiene le
       dimensioni dei fogli selezionabili
        */
42    struct sheets sheet[6] = {
43      {"a0paper", 118.9, 84.1}, {"
            a1paper", 84.1, 59.4},
44      {"a2paper", 59.4, 42}, {"a3paper",
             42, 29.7},
45      {"b1paper", 100, 70.7}, {"b2paper",
             70.7, 50}};
46    FILE *table;
47  /* Errore sul primo parametro. Uscita
       con codice 1 */
48    if ((pattern != 'T') && (pattern !=
          'I')) {
49      printf ("Tipo␣sconosciuto.␣Sono␣
            ammissibili␣solo␣T␣(testo)␣e␣I
            ␣(immagine)\n");
50      exit (1);
51    }
52    strcpy(sheetType, argv[2]);
53    strcpy(realPattern, argv[5]);
54    srand (time (NULL));
55  /* Recuperiamo le dimensioni del
       foglio per il successivo calcolo
       delle dimensioni reali delle celle
       . Se il formato è sconosciuto, il
       programma termina con codice 1 */
56    if (!strcmp (sheetType, "a0") || !
          strcmp (sheetType, "A0")) nSheet
          =0;
57    if (!strcmp (sheetType, "a1") || !
          strcmp (sheetType, "A1")) nSheet
          =1;
58    if (!strcmp (sheetType, "a2") || !
          strcmp (sheetType, "A2")) nSheet
          =2;
59    if (!strcmp (sheetType, "a3") || !
          strcmp (sheetType, "A3")) nSheet
          =3;
60    if (!strcmp (sheetType, "b1") || !
          strcmp (sheetType, "B1")) nSheet
          =4;
61    if (!strcmp (sheetType, "b2") || !
          strcmp (sheetType, "B2")) nSheet
          =5;
62    if (nSheet == 9) {
63      printf ("Formato␣di␣pagina␣
            sconosciuto.\n");
64      exit (1);
```

```
  }
  cellWidth = sheet[nSheet].sheetWidth / nCols;
  cellHeight = sheet[nSheet].sheetHeight / nRows;
  for (i = 0; i < nRows; i++)
    for (j = 0; j < nCols - 1; j++)
      strcpy (tabular[i][j], "␣&");
  for (i = 0; i < nRows; i++)
    strcpy (tabular[i][nCols - 1], "␣\\\\");
  for (i = i % 2; j < nCols; j += 2) {
    rotation = ((int) rand() % MAXROTSTEP ) * DEGSTEP;
    dimpos = (int) rand() % NUMDIM;
/* A seconda che il pattern di stampa sia testo o immagine, il vettore delle dimensioni conterrà dei comandi LaTeX o dei fattori di scala. Le dimensioni della minipage e della vbox sono quelle calcolate in precedenza a partire dalla dimensione del foglio e dal numero di righe e colonne */
    if (pattern == 'T') {
      char *dimen[NUMDIM] = {"\\large", "\\Large", "\\LARGE", "\\huge", "\\Huge"};
      sprintf (tmp, "\\begin{minipage}{%fcm}\\vbox␣to␣%fcm{\\vfill\\hfil\\rotatebox[origin=c]{%d}{%s␣%s}\\hfill\\vfill}\\end{minipage}␣%s", cellWidth, cellHeight, rotation, dimen[dimpos], realPattern, tabular[i][j]);
    }
    else {
      float dimen[NUMDIM] = {.75, .875, 1, 1.125, 1.25};
      sprintf (tmp, "\\begin{minipage}{%fcm}\\vbox␣to␣%fcm{\\vfill\\hfil\\rotatebox[origin=c]{%d}{\\includegraphics[scale=%f]{%s}}\\hfill\\vfill}\\end{minipage}␣%s", cellWidth, cellHeight, rotation, dimen[dimpos], realPattern, tabular[i][j]);
    }
    strcpy (tabular[i][j], tmp);
  }
  if ((table = open_file (fg, "w")) == NULL) exit (1);
  for (i = 0; i < nRows; i++)
    for (j = 0; j < nCols; j++)
      fprintf (table, "%s", tabular[i][j]);
  fclose (table);
/* Prepara il comando di shell per sostituire i dati relativi al foglio nel documento LaTeX */
  sprintf (tmp, "sed␣-i␣giftpaper.tex␣-e␣'7s/..paper/%s/g'", sheet[nSheet].latexname);
  system (tmp);
/* Prepara il comando di shell per sostituire i dati relativi alle abbondanze nel documento LaTeX */
  sprintf (tmp, "sed␣-i␣giftpaper.tex␣-e␣'8s/width=[0-9]*[.]*[0-9]*cm,height=[0-9]*[.]*[0-9]*cm/width=%.1fcm,height=%.1fcm/g'", sheet[nSheet].sheetWidth + .4, sheet[nSheet].sheetHeight + .4);
  system (tmp);
/* Prepara la stringa-comando per aggiornare il numero di colonne del file .tex */
  sprintf (tmp, "sed␣-i␣giftpaper.tex␣-e␣'14s/{[0-9][0-9]*}/{%d}/g'", nCols);
/* ... e la esegue */
  system (tmp);
  system ("pdflatex␣giftpaper");
  return 0;
}

FILE * open_file (char *nomefile, char *accesso)
{
  FILE *fp;

  if ((fp = fopen (nomefile, accesso)) == NULL)
    fprintf (stderr, "Errore␣nell'apertura␣di␣%s\n", nomefile);
  return fp;
}

/* Restituisce il valore massimo tra due interi */
int max (int x, int y)
{
  if (x > y) return x;
  return y;
}
```

Per ottenere il risultato della figura 4, ma col pattern ovviamente diverso per via della casualità, ci basta dare il comando

```
makeGP T A3 16 20 \\GuIT*
```

A questo punto ci fermiamo, ma la fantasia potrebbe scatenarsi ulteriormente: un vettore potrebbe contenere due o più nomi di immagine e uno

o più testi, da selezionare casualmente allo stesso modo delle dimensioni; si potrebbero programmare i flag in input in modo da non essere più costretti a ricordarci la sequenza esatta da dare alla riga di comando; anche un *help* non sarebbe male; pur poco significativa, la possibilità di avere il foglio con orientazione *portrait* invece della *landscape* scelta inizialmente potrebbe essere una miglioria, così come lo sarebbe eliminare i nomi dei file *legacy* e renderli parametrici, cioè far scegliere dalla riga di comando il nome del documento LaTeX da compilare e quello della tabella da includere. Ma lo scopo dell'articolo non è certo questo. Ci siamo già spinti molto oltre e lo faremo di più, ma in altra direzione, nell'appendice A.

Rimane da vedere come fare la stessa cosa per le altre versioni. Partiamo da alcune osservazioni preliminari. Tutte le versioni "statiche" mostrano che le misure dell'abbondanza sono espresse in termini assoluti, così come le misure delle celle. Sarebbe un'ottima cosa se crop permettesse di specificare un formato di carta e un'abbondanza; purtroppo non lo permette, preferendo far inserire delle misure assolute. In makeGP.c abbiamo notato come dobbiamo far eseguire un comando che cambi le misure dell'abbondanza (righe 96–97) in giftpaper.tex in accordo col foglio scelto. Questa cosa, accettabile laddove il cambiamento sia automatico, non è una cosa molto intelligente da fare nelle versioni alternative: potremmo non ricordare a memoria né le dimensioni del formato scelto né le dimensioni dell'abbondanza. Allora occorre trovare una strategia che ci permetta di non inserire tali dati all'atto dell'inclusione di crop nelle versioni alternative a quella in C + LaTeX. Quando cambiamo il numero di righe e colonne da porre sulla tabella, dobbiamo ricordarci di cambiarne anche l'intestazione. Farlo a mano può essere fonte di dimenticanza. Quindi vedremo come fare per cambiare meno cose possibili nelle versioni in LuaLaTeX, LaTeX e LaTeX3. Partiamo dalla versione LuaLaTeX.

```
\documentclass[12pt]{article}

\usepackage{fontspec}
\usepackage{guit}
\usepackage{tabu}
\usepackage[a3paper,hmargin=0pt,
    vmargin=0pt,landscape]{geometry}
\newdimen\cropwidth
\newdimen\cropheight
\cropwidth=\paperwidth
\cropheight=\paperheight
\advance\cropwidth by 4mm
\advance\cropheight by 4mm
\usepackage[center,width=\the\
    cropwidth,height=\the\cropheight]{
    crop}
\newcount\Trows
\Trows=16
\newcount\Tcols
\Tcols=20
\usepackage{xcolor}
\usepackage{graphicx}
\usepackage{luacode}

\begin{luacode}
local dimen = {"\\large", "\\Large",
    "\\LARGE", "\\huge", "\\Huge"}
local Trows = tex.count['Trows']
local Tcols = tex.count['Tcols']
local anglestep = 45
local pw = tex.dimen['paperwidth']
local ph = tex.dimen['paperheight']
local mt = {}
function randrot()
  tex.print(math.random(0,7) *
      anglestep)
end
function randimen()
  tex.print(dimen[math.random(5)])
end
function pwidth()
  tex.print(pw / Tcols .. "sp")
end
function pheight()
  tex.print(ph / Trows .. "sp")
end
function maketable()
  for i = 0, Trows - 1 do
    mt[i] = {}
    for j = 0, Tcols - 2 do
      mt[i][j] = " &"
    end
  end
  for i = 0, Trows - 1 do
    mt[i][Tcols - 1] = " \\\\"
  end
  for i = 0, Trows - 1 do
    for j = i % 2, Tcols - 1, 2 do
      mt[i][j] = "\\cell" .. mt[i][j]
    end
  end
  tab = io.open ("table.tex", "w")
  for i = 0, Trows - 1 do
    for j = 0, Tcols - 1 do
      tab:write (mt[i][j])
    end
  end
  tab:close ()
end
\end{luacode}

\newcommand\cell{\begin{minipage}{\
    directlua{pwidth()}}\vbox to \
    directlua{pheight()}{\vfill\hfil\
    rotatebox[origin=c]{\directlua{
    randrot()}}{\directlua{randimen()
```

```
    }\GuIT*}\hfill\vfill}\end{minipage
    }}
68  \begin{document}
69  \pagecolor{black!5!yellow!30}
70  \directlua{maketable()}
71  \noindent\begin{tabu} to \textwidth {@
        {}*{\Tcols}X}
72  \input{table}
73  \end{tabu}
74  \end{document}
```

Questa versione, presente nei sorgenti allegati col nome di `giftpaperluaDI.tex`, sembra notevolmente diversa dalla versione realizzata per generare un foglio A3. Discutiamone le differenze.

Per prima cosa dobbiamo far sì che l'inclusione del pacchetto crop (riga 13) sia indipendente dal foglio che specificheremo a geometry. Per farlo, ci basta definire due variabili di tipo dimensione (righe 7 e 8) a cui assegnare come valore le dimensioni del foglio definite in geometry (righe 9 e 10) aumentate del valore dell'abbondanza (righe 11 e 12) e usare queste variabili nell'includere crop. Quindi bisogna definire due contatori per il numero di riga e di colonna. Nella versione "statica" avevamo scritto questi valori in due variabili locali di Lua. In questo caso non possiamo perché non è consentito usare una funzione Lua nel preambolo, dunque non possiamo comunicare tali valori a LaTeX. Il passaggio attraverso `\newcount` (righe 14–17) è obbligatorio per poter rendere accessibili questi valori alle funzioni Lua senza doverli duplicare esplicitamente nel sorgente. Ora cambia la sintassi d'uso del contatore nell'intestazione della tabella (riga 71), ma è irrilevante: a noi interessa sapere di dover modificare la sola riga 6 per cambiare il foglio e modificare le sole righe 15 e 17 per modificare il numero di righe e di colonne da mettere nella tabella. Ovviamente, poiché le dimensioni delle celle saranno ricalcolate a ogni compilazione, ci serviranno due funzioni Lua che eseguano questo calcolo. Tali funzioni, `pwidth()` (righe 36–38) e `pheight()` (righe 39–41), non fanno altro che emettere rispettivamente un valore dato dal rapporto tra la larghezza del foglio e il numero di colonne della tabella e tra l'altezza del foglio e il numero delle righe della tabella. Questo numero è adimensionale, ma a noi serve un numero con una dimensione ed è per questo che posponiamo a questi valori l'unità di misura (`sp`, lo *scaled point* di TeX, la misura da cui vengono derivate tutte le altre) tramite l'operatore di concatenazione (`..`). Il comando `\cell` (riga 67) richiamerà le due funzioni al posto dei valori *hard coded* della versione "statica".

In accordo con quanto visto per la versione LuaLaTeX, la versione LaTeX (inclusa nei sorgenti come `giftpaperlatexDI.tex`) diventa così:

```
1   \documentclass[12pt]{article}
2
3   \usepackage[T1]{fontenc}
4   \usepackage{guit}
5   \usepackage{tabu}
6   \usepackage[a3paper,hmargin=0pt,
        vmargin=0pt,landscape]{geometry}
7   \newdimen\cropwidth
8   \newdimen\cropheight
9   \cropwidth=\paperwidth
10  \cropheight=\paperheight
11  \advance\cropwidth by 4mm
12  \advance\cropheight by 4mm
13  \usepackage[center,width=\the\
        cropwidth,height=\the\cropheight]{
        crop}
14  \newcounter{Trows}\setcounter{Trows
        }{16}
15  \newcounter{Tcols}\setcounter{Tcols
        }{20}
16  \usepackage{xcolor}
17  \usepackage{graphicx}
18  \usepackage{lcg}
19
20  \makeatletter
21  \def\nloop#1{%
22    \def\nl@@p##1##2\repeat#1{%
23    \def##1{##2\relax\expandafter##1\fi}
          %
24     ##1\let##1\relax}%
25  \expandafter\nl@@p\csname nl@@p-\
        string#1\endcsname
26  }
27  \makeatother
28  \newcommand\dimension[1]{%
29    \ifcase #1\relax
30    \or \large
31    \or \Large
32    \or \LARGE
33    \or \huge
34    \or \Huge
35    \fi}
36  \newcounter{x}
37  \newcommand\cell{\begin{minipage}{\
        cellwidth}\vbox to \cellheight{\
        vfill\hfil\chgrand[first=0, last
        =7] \setbox1=\hbox{\rand}\hskip-\
        wd1\multiply\value{rand} by 45\
        value{x}=\value{rand} \rotatebox[
        origin=c]{\value {x}}{\chgrand[
        first=1, last=5]\setbox1=\hbox{\
        rand}\hskip-\wd1\dimension{\value{
        rand}}\GuIT*}\hfill\vfill} \end{
        minipage}}
38
39  \begin{document}
40  \newwrite\tempfile
41  \immediate\openout\tempfile=table.tex
42  \pagecolor{black!5!yellow!30}
43  \newdimen\cellwidth
44  \newdimen\cellheight
45  \setlength\cellwidth{\paperwidth}
```

```
\setlength\cellheight{\paperheight}
\divide\cellwidth by \theTcols
\divide\cellheight by \theTrows
\newcounter{nrows}
\newcounter{ncols}
\value{nrows}=0
\nloop\a
  \value{ncols}=0
  \nloop\b
    \ifodd\value{nrows}%
      \ifodd\value{ncols}\immediate\
          write\tempfile{\unexpanded{\
          cell}}\fi
    \else%
      \ifodd\value{ncols}\relax\else\
          immediate\write \tempfile{\
          unexpanded{\cell}}\fi
    \fi
    \stepcounter{ncols}
    \ifnum\value{ncols}<\value {Tcols
        }\immediate\write \tempfile{ &}
        %
  \repeat\b
    \immediate\write\tempfile {\
        unexpanded{\\}}\stepcounter{
        nrows}
  \ifnum\value{nrows}<\value{Trows}%
\repeat\a
\immediate\closeout\tempfile
\noindent\begin{tabu} to \textwidth {@
    {}*{\theTcols}X}
\input{table}
\end{tabu}
\end{document}
```

Le righe 43–48 mostrano il codice LaTeX per il calcolo delle dimensioni di ogni cella. Le dimensioni `\cellwidth` e `\cellheight`, usate nel comando `\cell`, dovranno contenere le dimensioni orizzontale e verticale della cella. Dopo averle create e inizializzate alla larghezza e all'altezza della carta, le dividiamo rispettivamente per il numero di colonne (celle orizzontali) e di righe (celle verticali). Queste ultime quantità le avevamo definite alle righe 14–15. Dunque, quando decideremo di far compilare un documento su un foglio diverso dall'A3 scritto nel sorgente, o con righe e/o colonne in quantità differenti, tutti i cambiamenti che faremo sono concentrati alle righe 6 (formato della carta) e 14–15 (numero di righe e colonne).

Infine la versione LaTeX3 (`giftpaper3DI.tex`):

```
\documentclass[12pt]{article}

\usepackage[T1]{fontenc}
\usepackage{guit}
\usepackage{tabu}
\usepackage[a3paper,hmargin=0pt,
    vmargin=0pt,landscape]{geometry}
\usepackage{xcolor}
\usepackage{graphicx}
\usepackage{expl3}

\ExplSyntaxOn
\dim_new:N\cropwidth\dim_set:Nn\
    cropwidth{\dim_eval:n{\paperwidth
    + 4mm}}
\dim_new:N\cropheight\dim_set:Nn\
    cropheight{\dim_eval:n{\
    paperheight + 4mm}}
\usepackage[center,width=\the\
    cropwidth,height=\the\cropheight]{
    crop}
\int_new:N\Trows\int_set:Nn\Trows{16}
\int_new:N\Tcols\int_set:Nn\Tcols{20}

\cs_new:Npn \dimension #1
{ \if_case:w #1 \relax
    \or: \large
    \or: \Large
    \or: \LARGE
    \or: \huge
    \or: \Huge
  \fi}
\cs_new:Npn \cell
{\begin{minipage}{\cellwidth}\vbox to
    \cellheight{\vfill \hfil\rotatebox
    [origin=c]{\fp_eval:n {45*randint
    (0,7)}}{\dimension{\fp_eval:n{
    randint(1,5)}}\GuIT*}\hfill\vfill
    }\end{minipage}}
\ExplSyntaxOff
\begin{document}
\pagecolor{black!5!yellow!30}
\ExplSyntaxOn
\iow_new:N \tempfile
\iow_open:Nn \tempfile {table.tex}
\dim_new:N\cellwidth\dim_set:Nn\
    cellwidth{\dim_eval:n{\paperwidth
    /\Tcols}}
\dim_new:N\cellheight\dim_set:Nn\
    cellheight{\dim_eval:nv{\
    paperheight/\Trows}}
\int_new:N\nrows\int_set:Nn\nrows{0}
\int_new:N\ncols
\int_do_while:nNnn{\nrows}<{\Trows}{%
  \int_set:Nn\ncols{0}
  \int_do_while:nNnn{\ncols}<{\Tcols}{
      %
    \int_if_odd:nTF{\nrows}{%
      \int_if_odd:nTF{\ncols}{%
        \iow_now:Nn \tempfile {\cell}
      }{\relax}
    }{%
       \int_if_odd:nTF\ncols{\relax}{
           %
         \iow_now:Nn \tempfile {\cell}
       }%
    }
    \int_incr:N\ncols
```

```
51      \int_compare:nNnTF{\ncols}< {\
            Tcols}{%
52        \iow_now:Nn \tempfile{ &}
53      }{\relax}
54    }
55    \iow_now:Nn \tempfile {\\}
56    \int_incr:N\nrows
57  }
58  \iow_close:N \tempfile
59  \noindent\begin{tabu} to \textwidth {@
        {}*{\Tcols}X}
60  \input{table}
61  \end{tabu}
62  \ExplSyntaxOff
63  \end{document}
```

Come già detto per la versione "statica", anche questa versione è la pura traduzione del programma equivalente scritto in LaTeX, quindi non ripeterò la spiegazione che sarebbe una ripetizione. Come molti utenti di LaTeX3, posso senz'altro affermare, di aver apprezzato molto la coerenza delle interfacce delle funzioni di questa versione del linguaggio e, di conseguenza, la sua facilità d'uso.

10 Conclusioni e proposta

Il presente articolo ha mostrato alcuni metodi per ottenere della carta da regalo personalizzata con del testo o con un'immagine a piacere. I metodi mostrati sono eterogenei: C + LaTeX, LuaLaTeX, LaTeX e LaTeX3 in rappresentanza di tre filosofie: calcolare all'esterno tutti i valori da inserire in un documento LaTeX tramite un linguaggio non integrato (C) o integrato (LuaTeX), oppure calcolare detti valori internamente grazie a LaTeX(3). Ogni lettore può ricorrere alla tecnica che meglio gli si confà.

La proposta al Direttivo è quella di considerare l'adozione della carta da regalo personalizzata GUIT come gadget di benvenuto ai nuovi soci.

A Ricerche ottimizzatorie e perdite di tempo

La sezione 3 ha mostrato una possibile soluzione (contenuta nel file `makeGPtv1.c`) per risolvere il problema della produzione della carta regalo e, al margine, una possibile soluzione al riempimento a scacchiera di una tabella. Come sempre, ogni soluzione ha vantaggi e svantaggi. Per esempio, in passato non sempre il codice più leggibile era il più efficiente. Ora, cercare di ottimizzare un programma così banale è inutile: quante volte potrà mai essere eseguito? E quanto tempo occorre per un'esecuzione? Quindi, se pure si guadagnassero ben 2 ms, quale sarebbe il vantaggio? Specie se l'ottimizzazione porta via troppo tempo. Tuttavia tentiamo un'impossibile miglioria tenendo conto che i compilatori odierni sono talmente efficienti da rendere inutili le riscritture del codice più risapute.

Tabella 1: Rilevazione dei tempi medî di esecuzione di `makeGPtv1` (sezione 3).

Programma	Tempo utente (s)	Tempo totale (s)	cpu (%)
`time`[10]	0,242	0,242	—
`/usr/bin/time`[11]	0,235	0,236	99,6
`runtime`[12]	0,24	0,24	99

Nella trattazione che segue sono stati usati i seguenti programmi per rilevare i tempi: `time`, comando integrato di bash tanto da esserne parola riservata; `time`, la versione GNU del comando; `runtime`, utility descritta in Peek *et al.* (1997). Tutti i tempi sono stati rilevati sulla stessa macchina (Intel Core i7 con 16 GB di RAM) e nelle stesse condizioni (due terminali e `xpdf` attivi; su un terminale si rilevano i tempi di esecuzione, sull'altro, terminate le rilevazioni, si lancia `vi` per riportare i tempi rilevati). L'eseguibile da testare è stato preparato eliminando le righe di codice 35–40, cioè quelle relative alla scrittura della tabella su file e di compilazione del documento LaTeX. Dunque ci si concentra solo sull'efficienza del riempimento della tabella in memoria. L'esecuzione del file C così preparato dà come tempo di esecuzione zero, cioè inferiore al millisecondo. Per rendere più corpose le misure, il sorgente è stato modificato introducendovi un ciclo che ripete 10 000 volte le righe 22–34. I tempi sono quelli medî relativi a 25 esecuzioni del programma. La tabella 1 riporta i dati di esecuzione forniti dai tre "cronometri".

Si può notare che i tempi rilevati dai comandi di shell (`time` di bash e `runtime`, basato su csh e che misura i tempi singoli al millesimo ma dà la media al centesimo) sono coerenti tra loro, mentre quello di `time` di GNU dà risultati inferiori fino a un centesimo di secondo. Su `https://unix.stackexchange.com/questions/27920/why-bash-time-is-more-precise-then-gnu-time` si trova la seguente risposta:

> After some hardcore bash code examining I found out that bash `time`

10. Il comando fornisce i tempi fino al millisecondo e la media delle esecuzioni, calcolata con un programma esterno, è stata arrotondata alla terza cifra decimale. Non dà percentuali di CPU.

11. Il comando fornisce i tempi fino al centesimo e la media delle esecuzioni, calcolata con un programma esterno, è stata arrotondata comunque alla terza cifra decimale per avere un'idea del bilanciamento dei tempi di esecuzione. La percentuale di CPU è invece data senza decimali e il suo valor medio è stato arrotondato alla prima cifra decimale.

12. Lo script `runtime` sfrutta gli strumenti di csh per le rilevazioni e i calcoli. Lo *user time* rilevato arriva ai millesimi di secondo, ma il calcolo della media viene arrotondato ai centesimi di secondo. Il *total time* (user + system) è rilevato ed espresso in minuti + secondi + centesimi ma il valore medio riporta solo i minuti e i secondi. Infine, la percentuale di CPU è rilevata al primo decimale ma il calcolo del valor medio viene espresso senza decimali. Questa e le successive tabelle riporteranno il *total time*, pari allo *user time* quando il *system time* è 0, e la percentuale di CPU rilevata dallo script.

> uses `getrusage()` and GNU `time` uses `times()`. `getrusage()` is far more precise because of microsecond resolution.

Il perché di questa differenza di misurazione può probabilmente essere imputato all'effetto dei troncamenti dei valori rilevati. Neanche le informazioni riportate su https://stackoverflow.com/questions/12392278/measure-time-in-linux-time-vs-clock-vs-getrusage-vs-clock-gettime-vs-gettimeof attestano con chiarezza il motivo di tale differenza. Non essendo questo un argomento strettamente attinente al lavoro presentato, lascerò a quei lettori che abbiano necessità di farlo il compito di scoprire i motivi delle differenze di misurazione. Se tali lettori lo riterranno opportuno, potranno comunicare l'esito dei loro approfondimenti direttamente a me o in una futura comunicazione su ArsTEXnica.

Si possono migliorare le prestazioni della versione in C del programma? Ci si può certamente provare, sempre ricordando che stiamo tentando di limare millesimi a una versione che esegue 10 000 volte la stessa cosa, il che ci farà limare un decimillesimo di quei millesimi nel programma reale. Dunque, affrontiamo il compito con puro spirito didattico.

In prima istanza si può pensare che togliere l'operazione modulo (%) dal ciclo alla riga 29 possa aiutare. Per farlo, basta applicare la strategia adottata da due degli autori di Beccari *et al.* (2016): sdoppiare il ciclo per operare sulle righe dispari indipendentemente da quelle pari. Dunque, le righe 28–34 saranno sostituite da:

```
28    for (i = 0; i < NROWS; i += 2)
29      for (j = 0; j < NCOLS; j += 2) {
30        rotation = ((int) rand() %
              MAXROTSTEP ) * DEGSTEP;
31        dimpos = (int) rand() % NUMDIM;
32        sprintf (tmp, "\\begin{minipage
              }{2.1cm}\\vbox␣to␣1.856cm{\\
              vfill\\hfil\\rotatebox[
              origin=c]{%d}{%s\\GuIT*}\\
              hfill\\vfill}\\end{minipage}
              ␣%s", rotation, dimen[dimpos
              ], tabular[i][j]);
33        strcpy (tabular[i][j], tmp);
34      }
35    for (i = 1; i < NROWS; i += 2)
36      for (j = 1; j < NCOLS; j += 2) {
37        rotation = ((int) rand() %
              MAXROTSTEP ) * DEGSTEP;
38        dimpos = (int) rand() % NUMDIM;
39        sprintf (tmp, "\\begin{minipage
              }{2.1cm}\\vbox␣to␣1.856cm{\\
              vfill\\hfil\\rotatebox[
              origin=c]{%d}{%s\\GuIT*}\\
              hfill\\vfill}\\end{minipage}
              ␣%s", rotation, dimen[dimpos
              ], tabular[i][j]);
```

Tabella 2: Rilevazione dei tempi medî di esecuzione di `makeGPtv2` (appendice A).

Programma	Tempo utente (s)	Tempo totale (s)	cpu (%)
`time`	0,242	0,242	—
`/usr/bin/time`	0,233	0,233	99,5
`runtime`	0,24	0,24	99

```
40        strcpy (tabular[i][j], tmp);
41      }
```

Nei sorgenti allegati, il file in questione è `makeGPtv2.c` (ovviamente nella versione con scrittura della tabella su file e senza i 10 000 cicli inutili). Il doppio ciclo delle righe 28–34 si occupa di riempire le celle pari (0, 2, 4…) delle righe pari (0, 2, 4…); quello delle righe 35–41 riempie invece le celle dispari (1, 3, 5…) delle righe dispari (1, 3, 5…). Il codice è un po' più lungo e forse un po' più chiaro del precedente; il vantaggio nell'esecuzione (preparata con gli stessi criterî già descritti) è invece nullo, come si evince dalla tabella 2. Ciò induce a pensare che il compilatore esegua in autonomia lo sdoppiamento del ciclo.

Possiamo fare un altro tentativo di migliorare le prestazioni tenendo conto del fatto che il C memorizza gli array per riga in posizioni consecutive di memoria: tutti i dati della seconda riga di un array occuperanno le posizioni di memoria immediatamente successive a tutti i dati della prima riga dello stesso array. Questo, a costo di un po' di aritmetica dei puntatori aggiuntiva, permette di accedere a ogni posizione dell'array sommando gli indici all'indirizzo base dell'array anziché tramite gli operatori parentesi quadre. Lasciando inalterato il doppio ciclo (righe 28–34) della versione originale, modifichiamo solo le due righe seguenti:

```
32        sprintf (tmp, "\\begin{minipage
              }{2.1cm}\\vbox␣to␣1.856cm{\\
              vfill\\hfil\\rotatebox[
              origin=c]{%d}{%s\\GuIT*}\\
              hfill\\vfill}\\end{minipage}
              ␣%s", rotation, dimen[dimpos
              ], *(*(tabular + i) + j));
33        strcpy (*(*(tabular + i) + j),
              tmp);
```

che, come vediamo, sono meno comprensibili da chi abbia dimestichezza con gli accessi a un array solo tramite operatori parentesi quadre.

Tale modifica (file `makeGPtv3`) sembra essere "efficace" poiché i tempi di esecuzione sono leggeremente inferiori (tabella 3) rispetto agli altri casi.

Come ulteriore e ultimo tentativo, possiamo unificare il doppio ciclo. Come vedremo tra poco, questo comporta dei controlli un po' meno intuitivi per capire quando ci si trova su una riga pari o su una dispari. Innanzitutto ci serviranno tre ulteriori variabili di tipo `int`: le chiameremo `cells`, `rows`

Tabella 3: Rilevazione dei tempi medî di esecuzione di `makeGPtv3` (appendice A).

Programma	Tempo utente (s)	Tempo totale (s)	cpu (%)
`time`	0,239	0,24	—
`/usr/bin/time`	0,232	0,233	99,5
`runtime`	0,24	0,24	99

Tabella 4: Rilevazione dei tempi medî di esecuzione di `makeGPtv4` (appendice A).

Programma	Tempo utente (s)	Tempo totale (s)	cpu (%)
`time`	0,246	0,246	—
`/usr/bin/time`	0,242	0,243	99,6
`runtime`	0,24	0,24	99

e `cols`. Poi dobbiamo modificare dalla riga 28 del programma originale:

```
28    cells = NROWS * NCOLS;
29    for (i = 0; i < cells; i += 2) {
30      rotation = ((int) rand() %
            MAXROTSTEP ) * DEGSTEP;
31      dimpos = (int) rand() % NUMDIM;
32      sprintf (tmp, "\\begin{minipage
            }{2.1cm}\\vbox␣to␣1.856cm{\\
            vfill\\hfil\\rotatebox[origin=
            c]{%d}{%s\\GuIT*}\\hfill\\
            vfill}\\end{minipage}␣%s",
            rotation, dimen[dimpos], *(*(
            tabular) + i));
33      strcpy (*(*(tabular) + i), tmp);
34      rows = i / NCOLS;
35      cols = i % NCOLS;
36      if ((cols + 2) >= NCOLS) {
37        if (rows % 2) i--;
38        else i++;
39      }
40    }
```

Vediamo il funzionamento di questa versione (`makeGPtv4.c` nei sorgenti): calcola il numero di celle della tabella ed esegue un unico ciclo su tutte le celle saltandone una ogni due. Genera, come sempre, la rotazione e la dimensione, quindi memorizza la stringa LaTeX nella cella individuata da `*(*tabular) + i`, cioè la posizione iniziale in memoria dell'array aumentata del numero di righe per il numero di colonne (numero memorizzato in `i`) per il numero di byte della cella. Quest'ultima moltiplicazione è "gratuita" nell'aritmetica dei puntatori perché implicita nel tipo di variabile e dunque è calcolato direttamente dal C, come anche nel sorgente precedente in cui però gli indici rimanevano sdoppiati. Ora bisogna capire se siamo all'inizio di una riga e se questa è pari o dispari e avere un solo indice non ci aiuta. Dunque dobbiamo calcolare la riga e la colonna correnti a partire dall'indice `i` (righe 34–35) e chiederci se siamo prossimi alla nuova riga (l'ultima colonna di una riga è la 19ª; la prima iterazione arriverà alla 18ª prima di andare alla riga successiva, quindi bisogna chiedersi se il numero di colonna corrente +2 raggiunge o eccede il numero di colonne, nel qual caso si controlla se la riga è pari (e quindi ci si sposta in avanti di una cella) o dispari (caso per cui ci si sposta indietro di una cella). Non dimentichiamo mai che la prima riga e la prima colonna hanno indice 0 e sono considerate pari.

I maggiori controlli e calcoli di questa versione si riflettono sull'esecuzione nell'ordine dei millesimi (tabella 4). Pur nell'esiguità dell'aumento dei tempi, il buon senso sconsiglia di ricorrere a una tale soluzione anche per problemi di leggibilità e manutenibilità. Tutti questi esercizî di programmazione sono serviti a capire il vantaggio di un approccio rispetto agli altri e a rivedere alcuni concetti del linguaggio C o di ottimizzazione del codice (per esempio il *loop splitting* menzionato anche in Falk e Marwedel (2004, 2003)).

Allora ci arrendiamo qui? No, però ricorriamo alla forza bruta. Il compilatore usato in questi test è lo `gnu gcc`. Questo consente diversi livelli di ottimizzazione del codice. Col livello 3 o col livello *fast* si ottiene un tempo medio di esecuzione (calcolato da `runtime`) di 23 centesimi di secondo, cioè un centesimo in meno dei tempi precedenti.

Ciò conferma definitivamente quello che avevamo già detto all'inizio e spesso ripetuto: inutile tentare di migliorare il codice originale il cui numero di esecuzioni e tempo di esecuzione sono e saranno sempre esigui.

B Un confronto grezzo

Arrivati alla fine, e oltre, dell'articolo, come ci si può orientare verso la migliore soluzione? Stante che una soluzione migliore in assoluto non c'è, possiamo però fare alcune considerazioni per orientarci singolarmente verso una soluzione piuttosto che verso un'altra.

Prendiamo in considerazione alcuni fattori per ognuno dei quattro programmi da confrontare; ogni fattore orienterà un tipo di decisione: quante righe di codice compongono ogni programma?[13] Quanti linguaggi bisogna conoscere?[14] Quanti fi-

13. Il numero di righe di codice può non essere indicativo, perché in alcuni casi due o più comandi o istruzioni sono stati posti su un'unica riga e, a parte casi speciali, potremmo decidere di scrivere il programma su un'unica, lunga riga. Forse sarebbe più significativo il numero di comandi o istruzioni date. Ma diamo questo dato solo come "indice di tempo di stesura del programma", senza contare il tempo speso a ideare l'algoritmo e a codificarlo.

14. Questo dato è significativo soprattutto per chi non abbia interesse o necessità di conoscere linguaggi diversi da LaTeX.

le formano il programma?[15] Quant'è il tempo di esecuzione?[16]

La tabella 5 riassume tutti questi numeri. Il lettore ha la più completa libertà di interpretarli per capire se la soluzione che avrebbe scelto è veramente la migliore (per lui).

Come promesso, non discuterò i dati riportati nella tabella 5 ma vorrei porre l'attenzione mia e dei lettori su un dato apparentemente sconcertante: la versione C + LaTeX, pur lanciando una compilazione con LaTeX, risulta molto più veloce della compilazione diretta con LaTeX. La prima spiegazione che mi viene in mente è che, mentre la tabella generata dal programma C mette una riga della tabella su una riga del file, la tabella generata dal documento LaTeX mette su ogni riga del file una singola cella. Dunque la lettura del file-tabella è molto più lenta. Ma non dobbiamo dimenticare che anche i numeri casuali sono generati da un pacchetto di LaTeX e dalla sua aritmetica durante la compilazione del documento anziché da una libreria del C altamente efficiente prima che il documento sia compilato.

Se qualcuno, poi, si sorprenderà per la prestazione temporale di LuaLaTeX, potrà riflettere sul fatto che Lua è un linguaggio interpretato e non compilato.

C La versione di Egregio

Alla fine della scrittura dell'articolo, ho pensato di far leggere il frutto delle mie "fatiche" proprio all'Egregio Guru nel timore che le mie citazioni "spiritaffettuose" potessero risultare indisponenti. Non ho ricevuto alcuna osservazione in merito, ma in compenso ho ricevuto una lezione di programmazione senza pari.

Oltre ad avermi dato alcuni consigli per impaginare meglio l'articolo, mi ha mandato una versione "statica" in LaTeX3 che, pur "tradendo" il codice originale nel più puro spirito delle traduzioni, fa delle cose che solo una profonda conoscenza di TeX e una conoscenza di frontiera di LaTeX3[17] permettono di fare: riempie la tabella senza cicli `while` ma con le funzioni dedicate, e lo fa senza scrivere alcunché su file esterni. In più, usa una versione di `\int_case` che permette di specificare i numeri per il confronto, per cui permette un'espansione più facile e intuitiva verso eventuali dimensioni del font più piccole di `\normalsize`. Bisogna ammettere che l'espressività del codice dell'Egregio Guru, paragonata all'espressività del mio, è la stessa di uno scritto di Umberto Eco paragonato a una pagina di astine (o un discorso di Woodstock con Snoopy).

15. Oltre che sul numero totale di righe di codice, questo fattore incide su come mettere in cllegamento i varî file. Non fanno parte del dato i file generati automaticamente, ma solo quelli scritti a mano.

16. Quello qui riportato è il tempo medio su 25 esecuzioni misurate col `time` di bash.

17. Non potete compilare questo documento con una TeX Live 2018 o precedente.

Riporto qui di seguito il codice, così come mi è arrivato (e incluso nei sorgenti come `giftpaper3egreg.tex`): tutti faremo bene a studiarlo nei minimi dettagli.[18]

```
\documentclass[12pt]{article}

\usepackage[T1]{fontenc}
\usepackage{guit}
\usepackage[a3paper,hmargin=0pt,
    vmargin=0pt,landscape]{geometry}
\usepackage[center,width=42.4cm,height
    =30.1cm]{crop}
\usepackage{xcolor}
\usepackage{graphicx}
\usepackage{expl3}

\ExplSyntaxOn
\cs_new_protected:Npn \cartaregalo_
    size:n #1
{
 \int_case:nn { #1 }
  {
   {1}{\large}
   {2}{\Large}
   {3}{\LARGE}
   {4}{\huge}
   {5}{\Huge}
  }
}
\cs_new_protected:Npn \cartaregalo_
    cell:
{
 \begin{minipage}{2.1cm}
 \leavevmode\vbox to 1.856cm
  {
   \vss
   \hbox to 2.1cm
    {
     \hss
     \rotatebox[origin=c]{\fp_eval:n
         {45*randint(0,7)}}
      {
       \cartaregalo_size:n {\fp_eval:n
           {randint(1,5)}} \GuIT*
      }
     \hss
    }
   \vss
  }
 \end{minipage}
}
\ExplSyntaxOff

\begin{document}
\pagecolor{black!5!yellow!30}
```

18. Posso affermare che la visione di tale sorgente, ispirandomi una citazione cólta, mi ha riportato alla mente un apoftegma attribuito ad Alvaro Vitali: «davanti all'arte togliti le mutande e mettile da parte» (Aa.Vv., 1996, p. 74).

Tabella 5: Confronto numerico dei quattro programmi "statici".

Programma	Nº file	Nº Linguaggi	Righe di codice	Tempo di esecuzione (s)
C + LaTeX	2	2	51 + 17	0,354
LuaLaTeX	1	2	56	0,977
LaTeX	1	1	58	0,549
LaTeX3	1	2	59	0,696

```
\ExplSyntaxOn
\tl_new:N \l_cartaregalo_body_tl
\int_const:Nn \c_cartaregalo_trows_int
    { 16 }
\int_const:Nn \c_cartaregalo_tcols_int
    { 20 }
\int_step_inline:nn { \c_cartaregalo_
   trows_int }
{
 \int_step_inline:nn { \c_cartaregalo_
    tcols_int }
  {
   \int_if_odd:nTF { #1 }
    {
     \int_if_odd:nT { ##1 }
      {
       \tl_put_right:Nn \l_cartaregalo
           _body_tl { \cartaregalo_cell
           : }
      }
    }
    {
     \int_if_odd:nF { ##1 }
      {
       \tl_put_right:Nn \l_cartaregalo
           _body_tl { \cartaregalo_cell
           : }
      }
    }
   \int_compare:nT { ##1 < \c_
       cartaregalo_tcols_int }
    {
     \tl_put_right:Nn \l_cartaregalo_
         body_tl { & }
    }
  }
 \tl_put_right:Nn \l_cartaregalo_body_
    tl { \\ }
}
\setlength{\tabcolsep}{0pt}
\noindent\begin{tabular*}{\textwidth}{
   @{\extracolsep{\fill}}*{20}{c}@{}}
\tl_use:N\l_cartaregalo_body_tl
\end{tabular*}
\ExplSyntaxOff
\end{document}
```

Per un giusto confronto, il tempo medio di esecuzione del codice (sempre su 25 esecuzioni) è risultato 0,36 s: veramente un tempo notevole.

Ringraziamenti

Questo articolo è debitore di alcune persone. Iniziamo da Roberto Giacomelli e Luigi Scarso; il primo attivissimo nel divulgare LuaTeX, il secondo nello svilupparlo. È grazie al loro esempio che ho deciso di fare una "traduzione" del mio programma in LuaLaTeX. Poi Claudio Beccari: non solo l'occasione del regalo mi ha suggerito di preparare della carta da regalo speciale, ma un suo recente messaggio mi ha dato lo spunto finale per scrivere la versione in LaTeX3; non bastando ciò, si è prestato a leggere il draft definitivo dell'articolo a scatola chiusa e mi ha fornito diversi suggerimenti per migliorare tutti i codici, sia dal punto di vista dello sviluppatore, sia da quello dell'utente. Infine, Enrico Gregorio e il suo *alter ego* (inventato ma verosimile) Egregio Guru: lungi dal chiedergli lumi su qualunque cosa inerente LaTeX3, ho preferito studiarmi i suoi articoli e il suo pacchetto kantlipsum; le cocenti esperienze passate mi hanno consentito di prevedere e prevenire un paio di sue possibili osservazioni che ho scritto nell'articolo fingendo che me le abbia fatte lui (ma, per sua ammissione, non mi avrebbe consigliato lcg). Però la realtà è stata più incredibile della fantasia!

Riferimenti bibliografici

Aa.Vv. (1996). «Campagna abbonamenti». *Amarcord. Il lato oscuro del cinema*, (1).

Beccari, Claudio (2019). «Email del 27 e 28 ottobre 2019». Comunicazione privata.

Beccari, Claudio, Roberto Giacomelli e Maurizio Molinaro (2016). «Il concorso della scacchiera». *ArsTEXnica*, (25).

LuaTeX development team (2019). *LuaTeX Reference Manual*. www.luatex.org.

Falk, Heiko e Peter Marwedel (2003). «Control flow driven splitting of loop nests at the source code level». In *Proceedings of the Conference on Design, Automation and Test in Europe - Volume 1*. IEEE Computer Society, Washington, DC, USA, DATE '03. http://dl.acm.org/citation.cfm?id=789083.1022762.

— (2004). *Source Code Optimization Techniques for Data Flow Dominated Embedded Software*. Kluwer Academic Publishers, Boston.

Giacomelli, Roberto e Gianluca Pignalberi (2015). «Generare documenti LaTeX con diversi linguaggi di programmazione». *ArsTEXnica*, (20).

Kaps, Florian "Doc" (2018). «The magic of linotype: Save and restart a printing legend». https://www.kickstarter.com/projects/1755997589/the-magic-of-linotype-save-and-restart-a-printing/description.

Peek, Jerry, Tim O'Reilly e Mike Loukides (1997). *Unix Power Tools*. O'Reilly, Sebastopol. Reperibile online su https://docstore.mik.ua/orelly/unix/upt/ch39_04.htm (sezione 39-4, Runtime) e ftp://ftp.oreilly.com/published/oreilly/power_tools/unix/upt9707.tgz (sorgente).

Plazzi, Andrea e Edoardo Rosati (a cura di) (1996). *Al servizio dell'Eroe. Il Tex di Magnus*. Editrice PuntoZero, Bologna.

print24 (2019). «Carta regalo». https://print24.com/it/prodotti-di-stampa/carte-da-regalo/. Ultima consultazione: 18 novembre 2019.

Stack Exchange (2015). «Making a random number». https://tex.stackexchange.com/questions/212738/making-a-random-number.

The LaTeX3 Project (2019). *The LaTeX3 Interfaces*. Leggibile dando da terminale il comando `texdoc interface3`.

Il codice illustrato in questo articolo è disponibile, per i soci GUIT, a questo indirizzo: https://www.guitex.org/home/images/ArsTeXnica/codice/

▷ Gianluca Pignalberi
g dot pignalberi at gmail dot com