Il Resto del Pinguino - N° 2 - Luglio 2010 - Pubblicazione aperiodica della comunità italiana dei programmatori Gambas - www.gambas-it.org

Il Resto del Pinguino

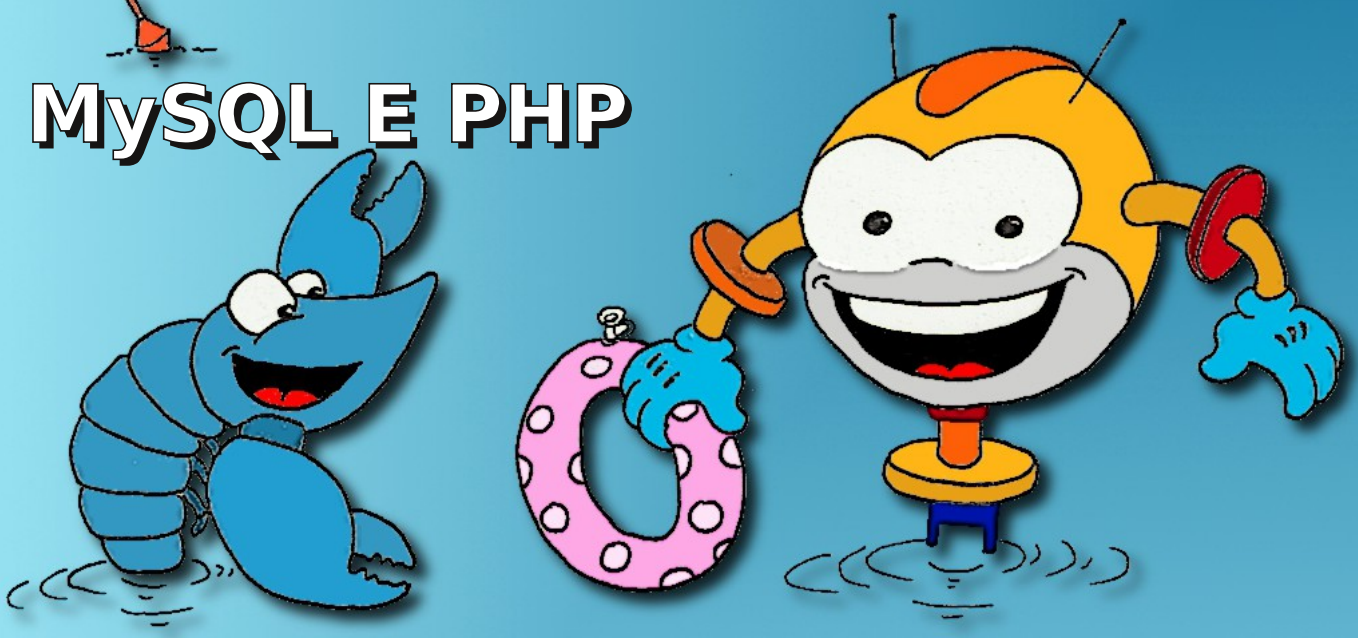
Open Source, Free Software e Programmazione

2

CRITTOGRAFIA FACILE CON GAMBAS!

TUTTE LE NOVITA' DI GAMBAS 3!

MySQL E PHP

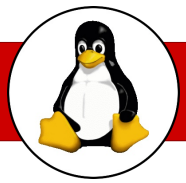


GUIDA AL "C"
Seconda puntata

Gesthotel
Gestionale alberghiero

WordPress 3.0
Tutte le novità

Editoriale



Numero estivo, questo, perché le vacanze sono alle porte, il caldo rallenta la mente e la voglia di fare. Ma non per questo abbiamo ridotto il nostro impegno, anzi, ed abbiamo deciso di lasciarvi con un numero bello peso! Peso non tanto per il numero di pagine ma per il contenuto. Abbiamo avuto la fortuna, infatti, di intervistare Fabien Bodard, uno dei componenti storici del team di sviluppo di Gambas, colui che segue lo sviluppo dell'IDE e di altri componenti. A Fabien, che ringrazio per la disponibilità e la gentilezza, abbiamo chiesto di anticiparci le novità che vedremo nella prossima versione di Gambas che dovrebbe essere ormai in dirittura di arrivo. Non solo, Fabien si è anche "sbottonato" rivelandoci uno dei componenti di punta del futuro Gambas 3.2. Ma ho già detto troppo e vi rimando alla lettura dell'intervista a pagina 4.

Abbiamo voluto anche giocare con la crittografia approfittandone contemporaneamente per una "lezione" sulle classi, i costrutti fondamentali della programmazione orientata agli oggetti. Il lettore sarà introdotto all'uso degli oggetti per strutturare il suo programma mediante la realizzazione di una classe crittografica capace di cifrare/decifrare un testo fornito in input. L'algoritmo scelto è l'RC4, inventato da Ron Rivest nel 1987. E' un algoritmo a flusso molto semplice per cui facilmente implementabile anche in Gambas. Sì, non è sicuro (è l'algoritmo responsabile delle violazioni delle reti wireless WEP) ma per i nostri scopi, didattici, presenta una sicurezza più che adeguata a proteggere dati non sensibili: l'importante è scegliere una buona password.

Una menzione la merita WordPress 3.0. La nuova versione di uno dei più diffusi blogger opensource introduce diverse novità facendo fare un discreto salto di qualità al software. Una importante novità è ora la possibilità di gestire, direttamente con WordPress, diversi blog con un'unica installazione dello stesso: un vero "plus" per chi gestisce diverse bacheche virtuali.

Infine, un grazie a tutti: il successo dei precedenti numeri ci ha spinto a continuare questa splendida avventura opensource con rinnovato vigore. Sinceramente l'apprezzamento per ciò che uno fa è il miglior stipendio che si possa intascare.

Leonardo "leo72" Miliani

REDAZIONE

Impaginazione: Leonardo "Leo72" Miliani - `leonardo@leonardomiliani.com`
Coordinazione articolisti: Francesco "Ceskho OpenCode" Apruzzese - `cescoap@gmail.com`
Grafica: Fabio "Pixel" Colinelli - `pixel.ubuntu@gmail.com`

COLLABORAZIONI

Se volete pubblicare un articolo sulla rivista, inviate la vostra opera (preferibilmente in forma di pacchetto compresso) all'indirizzo **`ilrestodelpinguino@gambas-it.org`** in formato testuale o, in alternativa, in formato ODT (OpenOffice), con eventuali foto come allegato. Nota: l'invio del materiale non obbliga la redazione alla sua pubblicazione. Sarà nostro insindacabile giudizio pubblicare o meno l'articolo ricevuto con eventuali tagli e modifiche dettati da motivi tipografici. Con l'invio del materiale l'autore dà tacito consenso all'utilizzo della sua opera ai fini della pubblicazione. Si ricorda che saranno pubblicati solo gli articoli inerenti i temi trattati dalla rivista.

Sommario

Il Resto del Pinguino

Num. 2 - Luglio 2010

4. News
Notizie e curiosità dal mondo dell'informatica

5. Fabien Bodard
Intervista allo sviluppatore dell'IDE di Gambas

7. RC4: crittografia con "classe"
Come creare una classe crittografica che usa l'algoritmo RC4

11. PHP e MySQL
Come recuperare dei dati da un database MySQL con PHP

13. Su che cosa e come realizzare un programma *(3ª parte)*
Il computer come intelligente strumento di sussidio alle attività umane

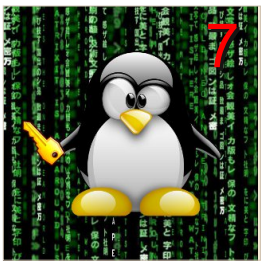
16. Guida al C in puntate *(2ª parte)*
Impariamo a programmare in C

18. Gesthotel
Gestionale alberghiero open-source

20. WordPress 3.0
Tutte le novità del blogger opensource più diffuso

21. La cattedrale ed il bazar *(3ª parte)*
Due stili di sviluppo del software a confronto: il modello "cattedrale" e quello "bazar"

Intervista a Fabien Bodard



Crittografia con Gambas

Gesthotel

News

Novità e curiosità dal mondo informatico

Adobe Flash player

Adobe blocca lo sviluppo del player **Flash** per le piattaforme GNU/Linux a 64 bit. Arriva così una notizia fulminante per gli utenti linuxiani frequentatori di siti carichi di contenuti Flash. La nota azienda ha escluso la versione di prova di Flash per 64 bit dal sito rendendo comunque avviabili quella per architetture a 32 bit. Tuttavia non si esclude un futuro sviluppo. Chiunque voglia continuare ad usare Flash su sistemi a 64 bit dovrà necessariamente rivolgersi a Google Chrome poiché Adobe sta conducendo lo sviluppo solo con questo browser (`http://www.adobe.com`).

Linux Mint 9 "Isadora"

Dal 18 maggio è disponibile anche **Linux Mint 9 "Isadora"**, la nuova versione del sistema operativo derivato da Ubuntu che da diverso tempo ormai è fissa nelle prime posizioni della classifica di gradimento degli utenti Linux (fonte: Distrowatch.com). Questo nuovo rilascio, basato su Lucid Lynx, introduce la trasparenza del Mint menu, un nuovo schermo di benvenuto e configurazione del sistema, il ritorno dell'installer per Windows, nuove grafiche per personalizzare il proprio desktop (`www.linuxmint.com`).

Ubuntu OS per tablet PC

Canonical sta lavorando ad una versione del suo sistema operativo Ubuntu pensato per i tablet PC, quella nuova generazione di computer portatili inaugurata da Apple con il suo iPad. Il sistema si dovrebbe chiamare Ubuntu Light ma, a differenza della versione Light che è attualmente in circolazione e che è destinata solo ai dispositivi *mobile* o *embedded*, sarà un mix fra la Netbook Edition ed una versione alleggerita della prossima versione di Ubuntu, la 10.10, che potrà essere utilizzata sia sui netbook che sui tablet. Punti forti del nuovo sistema saranno la leggerezza del software, il consumo energetico ed una interfaccia appositamente studiata per i touch-screen. (`www.networkworld.com`).

VLC 1.1.0

E' stata rilasciata la nuova major release di **VLC** che giunge così alla versione 1.1.0. L'ottimo player multimediale ora può avvalersi di nuove ed interessanti funzionalità e qualità. Abbiamo la decodifica GPU ed DSP, nuovo supporto per i codec, modulazione e demodulazione, estensioni LUA, rimozione di numerosi moduli, riscrittura del codice per migliorare l'uso delle risorse hardware ed altro ancora. Non resta che provare uno dei migliori software del panorama del software libero (`http://www.videolan.org/news.html`).

Slackware 13.1

Il 24 maggio è uscita una nuova versione di quella che può essere considerata come la più vecchia distribuzione Linux, **Slackware**. Fra le novità, l'utilizzo del kernel 2.6.33.4, la disponibilità delle ultime release di XFCE (4.6.1) e KDE (4.4.3), l'utilizzo di gcc in versione 4.4.4 e le ultime versioni dei software di gestione dei pacchetti di Slackare vale a dire slacktrack (utile per compilare i propri pacchetti) e slackpkg, che può anche aggiornare da una precedente versione di Slackware (`www.slackware.com`).

Crome OS ed il web

Il sistema operativo di Google, **Chrome OS**, sta lentamente prendendo forma. Non è atteso a breve (sarà disponibile per la fine del prossimo anno) ma ne sono state date alcune anticipazioni. Tutto ruoterà attorno a Chrome, il famoso browser della Casa di Mountain View, che sarà il tramite tra il sistema ed i programmi questi infatti saranno applicazioni web ed i documenti dell'utente e tutti i dati saranno memorizzati su web tramite un sistema di archivizione di tipo cloud. Il nocciolo del sistema sarà il kernel Linux ma Chrome OS sarà radicalmente differente dagli attuali sistemi: gli utenti dovranno dimenticarsi il concetto di desktop inteso alla maniera di Ubuntu o Suse, per intendersi. Il sistema servirà come substrato per avviare il browser (Google parla di 7 secondi), con il quale sarà possibile eseguire qualunque applicativo via web. Chrome OS non sarà scaricabile ma pre-installato in dispositivi quali netbook e simili (`www.networkworld.com`).

Fabien Bodard

Intervista allo sviluppatore dell'IDE di Gambas

In questo numero abbiamo il piacere di ospitare sulle nostre pagine Fabien Bodard, uno degli sviluppatori "storici" di Gambas, colui che segue l'IDE ed il componente gb.report. Vi invito a leggere l'articolo, soprattutto per scoprire le novità di Gambas 3. Buona lettura!

Il Resto del Pinguino: Ciao Fabien, potresti presentarti ai nostri lettori de "Il resto del Pinguino"?

Fabien: Il mio nome è Fabien Bodard, sono un semplice viticoltore, nella regione di Cognac. Perciò produco del favoloso Cognac :)

IRDP: Quali sono i tuoi interessi?

Fabien: Lo studio, la filosofia, le religioni (solo studio), la programmazione con Gambas ed altri linguaggi, la musica (sono un batterista e sto imparando a suonare il basso).

IRDP: Come hai conosciuto Linux? Su che distribuzione lavori?

Fabien: Ho conosciuto Linux nel 1996, perciò molto presto, grazie ad un libro intitolato "Linux secrets". Con questo libro c'era un CD con una Slackware. 48 ore di compilazione dopo avevo il mio primo sistema Linux installato... e malamente configurato... non l'ho usato per 2 anni e poi, nel 1998, ho installato la mia prima RedHat, e poi una Mandrake nel corso degli anni. Adesso uso Kubuntu da 3 anni. E sto per seguire Laurent Carlier su ArchLinux.

IRDP: Com'è il tuo angolo del computer?

Fabien: Hummm.... veramente disordinato *(fig. 1)*

IRDP: Potresti mostrarci il tuo desktop?

Fabien: Sì, è un desktop normale :) *(figg. 2 e 3)*. Nell'immagine c'è qualcosa che non esisterà fino a Gambas 3.2, il componente gb.g3d... è un'alpha e ad una fase di sviluppo molto iniziale ed è basato su Ogre. Un lavoro di Laurent Carlier.

IRDP: Potresti dirci qualcosa di più su questo componente?

Fabien: L'obiettivo è quello di avere qualcosa di simile a Blitz3D. Un sistema per usare gli oggetti e le scene 3D in modo realmente facile. Tu disegni la scena e gli oggetti 3D in un programma di modellazione, dove imposti anche il movimento, e poi chiami l'oggetto ed esegui i differenti movimenti attraverso alcune funzioni. Gambas serve giusto per gestire il gioco ed impostare le posizioni degli oggetti. Tutte le collisioni e le animazioni della scena sono gestite dal componente così che la parte interpretata non renda il gioco troppo lento.

Ma è un componente futuro come quello che gestirà la libreria ncurses, che permetterà di realizzare applicazioni da terminale complete, non sono per Gambas 3 ma minimo per Gambas 3.2

IRDP: Ora una domanda che interessa tutti gli utenti Gambas: come hai conosciuto Gambas, e quali sono i tuoi compiti all'interno del progetto?

Fabien: Il mio incontro con Gambas? Ahhh, è una grande storia :-)

Nel maggio del 2002 ero ad un mercato a vendere le mie meravigliose bottiglie e leggevo, mentre aspettavo i clienti, una rivista di computer, 'LOGIN'. Questa rivista era orientata ai sistemi di programmazione free e c'era un piccolo inserto dedicato a 'Gambas'. Da lì, caricai questo interprete con IDE integrato, e mi divertii con esso. Poi contattai Benoit Minisini per ringraziarlo per quello splendido progetto. Iniziai disegnando il logo della versione 1; poi mi sono dedicato all'IDE, ho tradotto la documentazione, ed aiuto i nuovi utenti sulla mailing-list.

IRDP: Gambas è un progetto che ruba molto del tuo tempo libero?

Fabien: Sì, è sicuramente così.... chiedete alla mia fidanzata.

IRDP: Ci sono un sacco di voci su Gambas 3: potresti per favore anticipare ai nostri lettori cosa c'è di nuovo in Gambas 3 e quali sono le differenze rispetto a Gambas 2?

Fabien: Le basi del linguaggio rimangono le

Il nuovo logo di Gambas 3

stesse, fortunatamente. Fanno la loro comparsa le strutture. E possiamo produrre tabelle per tutti i tipi di variabile o classe.

```
Dim aClass = NEW MyClass[]

  Public Struct <StructName>
    <Var1> As <Type1>
    <Var2> As <Type2>
  End Struct
```

C'è anche un nuovo modulo di disegno che permette di creare grafica vettoriale con anti-aliasing: questa classe può anche gestire file in formato SVG e PDF e stampare.

Le classi gb.qt4 e gb.gtk sono ora simili come non mai, con, ad esempio, il supporto per la stampa da GTK. C'è anche una libreria per usare Webkit di Qt4, ed un'altra per connettersi a DBus, ma manca ancora della gestione dei segnali.

Anche l'IDE è molto migliorato: fornisce assistenza sulla digitazione mediante una nuova struttura di completamento del codice per accelerarne la scrittura, ed il browser interno può adesso accedere direttamente alla documentazione.

La libreria gb.smtp supporta l'autenticazione.

Gb.mysql supporta pienamente MySQL; gb.sdl supporta l'accelerazione grafica (Laurent Carlier ha riscritto completamente il codice).

La libreria gb.report è stata completamente riscritta e l'IDE permette adesso gli stati di stampa, ma a differenza della tecnica utilizzata da altre soluzioni di stampa, essa utilizza lo stesso sistema dei form grafici. Quindi abbiamo un contenitore e delle proprietà, poi noi passiamo un Risultato o una Collezione ed il riempimento è fatto dalla libreria stessa.

E' adesso possibile manipolare le immagini nella modalità terminale.

Per finire, e senza finire veramente perché ho tralasciato un sacco di cose, il Compilatore e l'Interprete sono stati velocizzati enormemente.

E scusami per le notizie messe tutte insieme :)

IRDP: Adesso una domanda da un milione di dollari: potremmo sapere quando sarà rilasciato?

Fabien: Quando avrò finito la struttura base del componente gb.report (sono lento...).

IRDP: Grazie per il tempo che ci hai dedicato XD

Leonardo "leo72" Miliani

Figura 1: l'angolo computer di Fabien

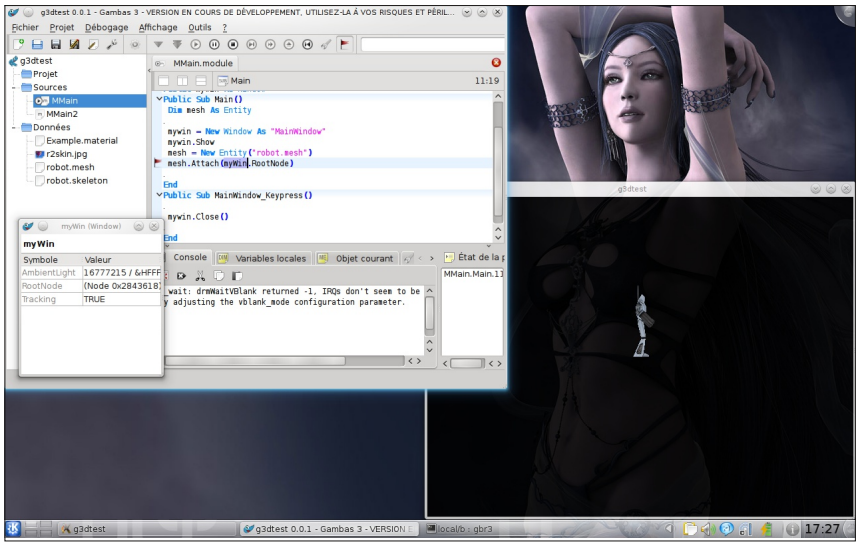


Figura 2: il desktop di Fabien. Notare il nuovissimo componente g3d...

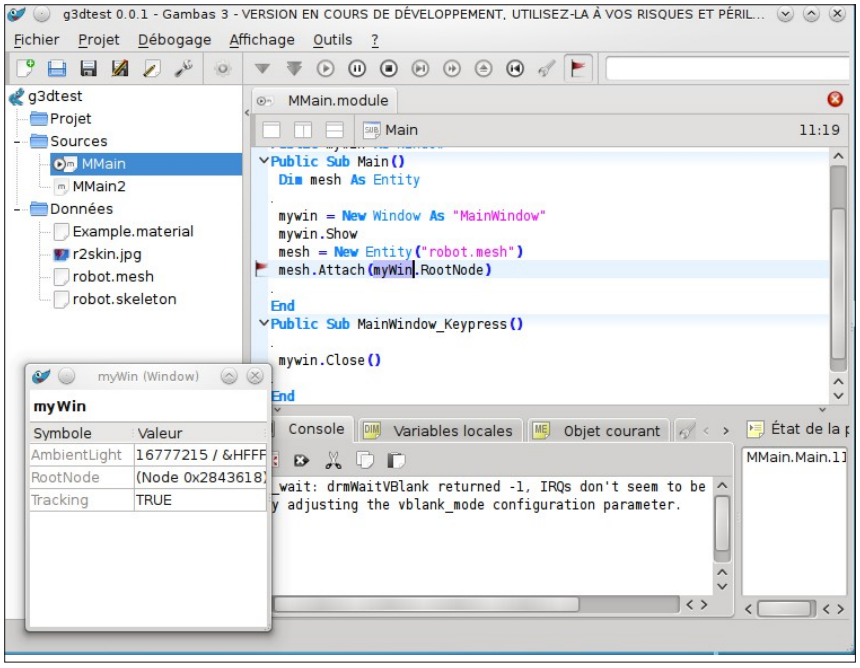


Figura 3: ...che farà la sua comparsa nel futuro Gambas 3.2

RC4: crittografia con "classe"

Come creare una classe crittografica che usa l'algoritmo RC4

Crittografia è una parola che deriva dal greco e significa “scrittura nascosta”: come si evince dal suo significato, la crittografia è la tecnica mediante la quale si cerca di rendere illeggibile (nascosto) uno scritto a chi non conosce il metodo o la chiave (o entrambi) utilizzati per cifrare il testo originale.

Com'è facile intuire, la crittografia ha beneficiato non poco dell'avvento dei calcolatori elettronici, evolvendosi dai metodi "carta-e-penna" alle sofisticate tecniche che sfruttano ampiamente l'informatica e la potenza delle macchine: è grazie ai computer, infatti, che sono nati tutti quegli algoritmi crittografici che sono oggi comunemente utilizzati per la protezione dei dati. Ad esempio, la connessione che sta "legando" il vostro portatile al *router* nella stanza accanto sfrutta un algoritmo crittografico per trasferire in sicurezza i pacchetti da e verso il computer e per evitare che qualche "furbetto" tenti di sfruttare la vostra connessione ad internet agganciandosi al vostro modem.

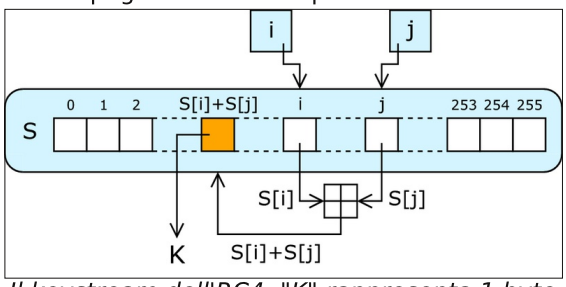
Ecco, l'oggetto di questo articolo è l'algoritmo che è stato utilizzato per cifrare le connessioni in due famosissimi standard wireless, il WEP ed il WPA prima versione: si tratta dell'**RC4**. Nonostante sia ritenuto insicuro, perciò inadatto ad un livello di sicurezza elevato, l'algoritmo è molto semplice da implementare e rappresenta una buona "palestra" per far pratica con un concetto base della *programmazione ad oggetti*: le **classi**. Nell'articolo che segue vedremo cos'è l'RC4, come si può implementare una classe in Gambas e come utilizzare questa classe all'interno dei nostri programmi.

Un po' di storia

L'RC4 (acronimo di *Rivest Cipher 4*) è un algoritmo crittografico a flusso con chiave simmetrica, inventato da Ronald Rivest (RSA Security), nel 1987. Un cifrario a flusso è un algoritmo che cifra i dati man mano che questi vengono resi disponibili; la cifratura avviene in maniera indipendente per ogni singolo dato. Un cifrario a chiave simmetrica è, invece, un tipo di algoritmo crittografico che utilizza la stessa chiave sia per l'operazione di cifratura che per quella di decifratura: l'alternativa sono i cifrari a chiave asimmetrica, che utilizzano due distinte chiavi per le suddette operazioni.

L'algoritmo rimase inizialmente segreto ma, nel 1994, fu reingegnerizzato e pubblicato su un newsgroup internazionale. Da allora si diffuse rapidamente perché facilmente implementabile e dotato di buone prestazioni. Semplificando, l'RC4 opera generando un flusso di dati pseudo-casuali (detto "*keystream*") che viene utilizzato per cifrare mediante un'operazione di OR esclusivo (XOR) i dati in chiaro: il *keystream* è generato con l'appoggio di una *S-box*, una funzione che sostituisce i caratteri in chiaro in modo da alterare eventuali relazioni riscontrabili fra il testo in chiaro e quello cifrato. L'algoritmo utilizza una chiave crittografica lunga da un minimo di 40 ad un massimo di 256 bit (da 5 a 32 caratteri), fornita dall'utente.

L'RC4 paga la sua semplicità in termini di



Il keystream dell'RC4: "K" rappresenta 1 byte generato dalla funzione

sicurezza tant'è che nel 2005 è stato messo a punto un metodo che viola una connessione WEP in meno di 1 minuto. Questo metodo sfrutta uno dei punti deboli dell'RC4: l'uso della stessa chiave crittografica porta a generare la stessa sequenza di bit cifrati in presenza dello stesso testo da cifrare. Siccome una connessione wireless utilizza la stessa chiave per tutta la durata della comunicazione e siccome l'implementazione dell'RC4 usato nel protocollo WEP utilizza una chiave lunga solamente 40 bit, forzare una tale connessione in presenza di un traffico regolare e costante non è difficile.

Ma a noi non interessa tutto ciò, dato che siamo utenti precisi e quindi utilizzeremo sicuramente delle buone chiavi crittografiche, composte da lettere minuscole e maiuscole, caratteri di punteggiatura e numeri, con una lunghezza della chiave più che sufficiente a tener segreti i nostri dati per qualche anno. Quindi, vediamo un po' come implementare l'RC4 in Gambas: per far ciò utilizzeremo una classe, cercando di capire cosa essa sia.

Le classi e la programmazione OO

In informatica una *classe* è un modello usato per creare degli oggetti: una classe è quindi un costrutto fondamentale della cosiddetta **programmazione OO**, o ***Orientata agli Oggetti*** (*Object Oriented*). Una classe è una porzione di codice che comprende *attributi* (variabili) e *metodi* (funzioni e procedure) che possono essere utilizzati da tutti gli oggetti che derivano da questa classe. Per derivare un oggetto da una classe predefinita non dobbiamo far altro che creare una "*istanza*" di tale classe, vale a dire una copia uguale ma indipendente alla classe di origine.

Per capire questo meccanismo, ipotizziamo una fabbrica di automobili. Gli ingegneri disegnano e progettano un nuovo modello, e forniscono il file grafico al sistema robotizzato della fabbrica di produzione. Poniamo che questo file sia chiamato "SuperSport". Nella fabbrica il sistema di costruzione analizzerà il file e creerà tante istanze, cioè tante "copie" del progetto originale, tutte uguali nella struttura del telaio e della carrozzeria, ognuna però differente dalle altre nei dettagli e nelle personalizzazioni: a seconda delle ordinazioni ricevute dai clienti, si dovrà magari produrre una SuperSport rossa con interni neri, oppure bianca con interni in pelle, con un motore 3000 cc, con pneumatici maggiorati, ecc... Ogni modello della SuperSport sarà quindi differente dalle altre copie prodotte.

Trasportando nel campo informatico il nostro esempio automobilistico, possiamo pensare al progetto originale come alla "classe SuperSport", ai modelli prodotti come alle "istanze" della classe SuperSport, alle personalizzazioni come agli "attributi" della classe. La classe originale avrà un attributo "motore", un attributo "pneumatici", un attributo "colore vernice", ecc....: questi attributi, durante la produzione, cioè durante l'istanziamento delle classi derivate, riceverenno un valore preciso. Nell'esempio avevamo un motore da 3000 cc. Quindi, scrivendo utilizzando la sintassi classica della programmazione OO con la quale si separano i metodi ed attributi di una classe mediante l'uso di un punto ".", creeremo un'istanza della classe per il cliente "Leo" con, ad esempio, questo codice:

```
SuperSportLeo = SuperSport.New()
```

Questa istruzione crea l'oggetto "SuperSportLeo" come una istanza della classe SuperSport: il metodo `New()` indica proprio all'interprete Gambas che deve creare una nuova copia della classe.

Adesso potremmo assegnare dei valori ai suoi attributi. Il cliente Leo ha deciso di acquistare un modello color rosso, con interni in pelle, motore di 3000 cc e cambio a 6 marce. Potremmo ipotizzare un'assegnazione come la seguente:

```
SuperSportLeo.motore = 3000
SuperSportLeo.colore = Colori.rosso
SuperSportLeo.cambio = Marce.sei
SuperSportLeo.interni = Interni.pelle
```

Qui vediamo come l'assegnazione di valori agli attributi di una classe si esegue come una normale assegnazione a variabili. `SuperSportLeo.motore = 3000` dice all'interprete che l'attributo "motore" della classe SuperSportLeo deve avere il valore di 3000.

Sotto abbiamo altri interessanti esempi di utilizzo di oggetti. Siccome i colori si possono immaginare come elementi di un unico insieme, anche per il colore utilizziamo una classe "Colori" contenente diversi attributi, ognuno ad indicare una particolare tonalità cromatica. L'interprete, quando vedrà che l'attributo `SuperSportLeo.colore` richiede il colore `Colori.rosso`, non farà altro che prelevare dall'attributo "rosso" della classe "Colori" il valore tintometrico della tonalità. Stesso discorso per l'attributo "cambio", che riceverà il valore dell'attributo "sei" della classe "Marce", e così come per gli interni, che saranno quelli stabiliti dal valore di `Interni.pelle`.

Dino è invece un guidatore più prudente ed ha ordinato la SuperSport solo perché amante della sua linea. Ha perciò scelto un modello meno performante e più ordinario: motore 2000 cc, colore grigio ed interni in tessuto. Ecco il suo ordine utilizzando la notazione della programmazione OO:

```
SuperSportDino = SuperSport.New()
SuperSportDino.motore = 2000
SuperSportDino.colore = Colori.grigio
SuperSportDino.cambio = Marce.cinque
SuperSportDino.interni=Interni.tessuto
```

La classe RC4

Siamo adesso pronti a trasferire quanto da noi appreso sull'uso della programmazione OO in una classe crittografica capace di cifrare i nostri messaggi in chiaro decifrare i messaggi crittati. Il codice dell'RC4 è molto semplice ed è pertanto facile farne una versione in Gambas.

La prima cosa da fare è quindi aprire un nuovo progetto Gambas, dargli un nome e poi cliccare con il tasto destro nell'area di sinistra e scegliere "Nuovo/Classe...". Chiamiamo questo nuovo oggetto **clsRC4** (il suffisso *cls* sta per classe e ci servirà per ricordare la natura del nuovo oggetto). La prima parte di codice che andremo a scrivere riguarderà la dichiarazione di alcune variabili globali che saranno usate nella classe. Specificatamente abbiamo bisogno di una ***S-Box*** e di un ***keystream***. L'*S-box*, come abbiamo detto in apertura, è una funzione che serve ad oscurare le relazioni tra il testo in chiaro e quello cifrato mediante una operazione di sostituzione dei caratteri (la "S" sta proprio per "*Substitution*"). Grazie alla *S-box* il nostro *keystream*, che è il flusso di dati da usare per cifrare quelli in chiaro,

risulterà più efficiente in termini di sicurezza. Queste variabili le dichiariamo nella parte iniziale della classe, così che siano accessibili ad ogni funzione o procedura della stessa.

```
PRIVATE s AS NEW Integer[]
PRIVATE kep AS NEW Integer[]
```

La seconda parte del codice riguarda le istruzioni che la classe esegue quando viene istanziata, cioè quando ne viene creata una nuova "copia". Per far ciò basta inserire il codice nel metodo _New della classe: questo metodo viene eseguito in automatico da Gambas quando, appunto, l'utente chiede all'interprete di generare una nuova classe derivata da quella scritta da noi.

```
PUBLIC SUB _new()

  s.Clear
  kep.Clear
  s.Resize(256)

END
```

Il codice è molto semplice: puliamo i vettori e li ridimensioniamo a 256 caratteri di lunghezza.

Adesso non ci resta che scrivere la routine che si occuperà del setup dell'algoritmo. Questa routine prepara il keystream e l'S-Box e deve essere chiamata passandogli come parametro la chiave di cifratura. Ad ogni chiave corrisponde un keystream ed una S-Box differente per cui solo usando la stessa chiave utilizzata per la cifratura è possibile decifrare il messaggio segreto e riottenere il testo in chiaro. Ecco il codice:

```
PRIVATE SUB preparaRC4(chiave AS String)
'-- inizializza l'algoritmo
DIM temp, a, b, j AS Integer

  '-- trasforma la chiave in una
  '-- matrice di byte (keystream)--
  b = 0
  FOR a = 0 TO 255
    b += 1
    IF b > Len(chiave) THEN
      b = 1
    END IF
    kep[a] = Asc(Mid$(chiave, b, 1))
  NEXT

  '-- inizializza la funzione S-box
  FOR a = 0 TO 255
    s[a] = a
  NEXT
  b = 0
  '-- scarta i primi 512 byte generati
  '-- (sono i più vulnerabili ad una crittoanalisi dell'RC4)
  FOR j = 1 TO 2
    FOR a = 0 TO 255
      b = (b + s[a] + kep[a]) MOD 256
      SWAP s[a], s[b]
    NEXT
  NEXT

END SUB
```

Analizziamo riga per riga:

1. dichiariamo alcune variabili che saranno usate nella procedura.

2. Il primo ciclo FOR..NEXT genera il keystream. Questo è lungo 256 caratteri ed è creato inserendo in un vettore i caratteri della chiave segreta ciclicamente, ripetendo l'inserimento finché non si è riempito tutto il keystream.

3. Adesso azzeriamo il contenuto dell'S-Box.

4. Ora scartiamo i primi 256 byte generati dal keystream perché presentano una certa periodicità, non da vero PRNG (Pseudo-random number generator). Siccome vogliamo fare le cose per bene, ripetiamo l'operazione 2 volte (scartiamo cioè 512 byte).

Adesso non ci resta che scrivere il codice riguardante l'algoritmo RC4 vero e proprio:

```
PUBLIC FUNCTION DeCifra(testo AS String, chiave AS String) AS String
DIM tempo, k, i, j AS Integer
DIM contatore, lunghezza AS Integer
DIM MatriceTemp AS NEW Byte[]
DIM stringa AS String

  i = 0
  j = 0
  preparaRC4(chiave)
  lunghezza = Len(testo)
  MatriceTemp.Clear
  FOR tempo = 1 TO lunghezza
    MatriceTemp.Add(CByte(Asc((Mid$(testo, tempo, 1)))))
  NEXT

  FOR contatore = 0 TO lunghezza - 1
    i = (i + 1) MOD 256
    j = (j + s[i]) MOD 256
    SWAP s[i], s[j]
    '-- genera la chiave k (in formato byte) --
    k = s[(s[i] + s[j]) MOD 256]
    '-- operazione di XOR fra un byte del testo
    '-- in chiaro ed uno della chiave --
    MatriceTemp[contatore] = MatriceTemp[contatore] XOR k
  NEXT
  stringa = ""
  FOR tempo = 0 TO lunghezza - 1
    stringa &= Chr(MatriceTemp[tempo])
  NEXT
  RETURN stringa

END FUNCTION
```

Esaminiamo il codice:

1. abbiamo deciso di utilizzare una funzione perché l'algoritmo deve accettare dei dati in ingresso e deve restituire una stringa in uscita, che è il testo de/cifrato.

2. Definiamo alcune variabili che saranno utilizzate dall'algoritmo.

3. Inizializzare il keystream e l'S-box (funzione preparaRC4).

4. Carichiamo in una matrice monodimensionale di tipo Byte, "MatriceTemp", i soingoli caratteri del testo passato come argomento.

5. Inizia il processo di elaborazione vero e

proprio: ogni singolo elemento di MatriceTemp verrà combinato mediante uno XOR con un elemento dell'*S-box*. Da notare come quest'ultimo non viene scelto linearmente dall'*S-box* ma viene selezionato mediante una operazione un po' complessa, in cui entra in gioco anche uno scambio di elementi dell'*S-box* stessa. Tutte queste operazioni servono per eliminare la "linearità" dall'algoritmo, così da renderlo più difficilmente analizzabile.

6. Terminata l'esecuzione dell'algoritmo RC4 viene ricostruita una stringa con tutti i caratteri di MatriceTemp, che è poi restituita come risultato della funzione.

Come usare la classe RC4

Fin qui abbiamo visto come creare una classe. Adesso utilizziamola nei nostri programmi. Quello che dobbiamo fare è semplicemente creare un'istanza della classe originale ed utilizzarla come un qualunque altro oggetto di Gambas. Ecco il codice:

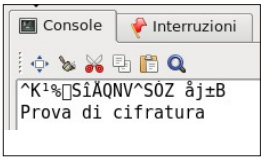
```
DIM RC4 AS NEW clsRC4
DIM Chiave, Testo, Testo2, TestoDaProcessare AS String

  Testo = "Prova di cifratura"
  Chiave = "Chiave"
  Testo2 = RC4.DeCifra(Testo, Chiave)
  PRINT Conv(Testo2, "ISO-8859-1", "UTF-8")
  Testo = RC4.DeCifra(Testo2, Chiave)
  PRINT Testo
```

La prima riga è quella che più ci interessa: dichiara un oggetto di nome RC4 (`"DIM RC4"`) come istanza, cioè copia, della classe *clsRC4* (`"AS NEW clsRC4"`). Con questa semplice riga Gambas crea la classe figlia RC4 che deriva dalla classe genitore clsRC4, da cui eredita tutti i suoi metodi. Il punto focale è l'ultima parte della riga: la parola `NEW` seguita dal nome della classe genitore. `NEW` indica all'interprete che deve creare un nuovo oggetto, non un riferimento. Quindi, RC4 è in tutto e per tutto indipendente ma identico a clsRC4.

La seconda riga di dichiarazioni non è molto importante: serve solo a creare delle variabili utilizzate nell'esempio, a cui assegniamo un testo da cifrare, la chiave segreta, ed i risultati delle operazioni.

Eccoci al punto cruciale, l'uso della nostra classe crittografica. Come possiamo vedere l'uso è semplice: basta utilizzare la classica sintassi *oggetto.metodo* valevole per qualsiasi altro oggetto di Gambas. Nello specifico, basta scrivere `RC4.DeCifra(Testo, Chiave)` ed assegnare il risultato ad una variabile. È interessante notare che l'editor di codice "sa" che RC4 è un'istanza di clsRC4 grazie alla dichiarazione che abbiamo fatto prima per cui ci assisterà durante la digitazione suggerendoci i metodi della classe non appena scriviamo il carattere ".", metodi che RC4 ha ereditato, come detto, da clsRC4.

Ora stampiamo sul terminale di Gambas il risultato della cifratura del testo: utilizziamo la funzione `Conv()` per convertire il risultato ottenuto dalla classe, in formato UTF-8, in un formato stampabile a video. Poi facciamo l'operazione inversa, riconvertiamo cioè il testo cifrato in testo in chiaro: come abbiamo detto all'inizio dell'articolo, l'RC4 è un algoritmo crittografico a chiave simmetrica, vale a dire che utilizza la stessa chiave sia per cifrare che per decifrare. Quindi per ottenere il testo originale non dobbiamo far altro che passargli il testo cifrato con la stessa chiave utilizzata in precedenza, come si vede nella penultima riga di codice dell'esempio. Stampiamo poi il risultato sul terminale per verificare che l'operazione è stata eseguita correttamente: se tutto è andato per il verso giusto (se cioè non abbiamo sbagliato a scrivere il nostro codice), vedremo apparire una prima incomprensibile riga di testo (la versione cifrata del nostro messaggio) seguita dal testo originale, versione decifrata della precedente. Questa è la conferma che la nostra classe crittografica funziona correttamente!



Nel terminale devono comparire queste due righe: la versione cifrata e quella in chiaro del nostro messaggio.

Conclusioni

Grazie alla crittografia ci siamo divertiti a studiare come utilizzare gli oggetti nella programmazione di "tutti i giorni": non è vero che gli oggetti servono solo in determinate situazioni o per determinati scopi. E' proprio questa convinzione, sommata all'idea molto diffusa che la programmazione OO sia macchinosa e difficile, che molti programmatori non utilizzano gli oggetti nei propri sorgenti. E' vero, avremmo potuto rendere la classe RC4 con una funzione: però in questo modo avremmo dovuto “annegare” tale funzione nel nostro codice e non avremmo avuto nessun controllo su di essa. Nel caso in questione, questo punto non è molto importante, ma pensate ad un oggetto diverso, ad esempio un contenitore di funzioni matematiche. Ogni funzione che avremmo voluto aggiungere altro non sarebbe stato che un nuovo metodo del nostro oggetto. Se invece avessimo utilizzato le classiche funzioni, ogni nuova funzione sarebbe stata una porzione di codice a sé stante: alla fine avremmo avuto decine di funzioni ognuna staccata dalle altre. Con un oggetto che le racchiude tutte, basta un "." ed esse sono tutte lì, a disposizione ed in buon ordine.

Infine, una classe esterna è anche un ottimo modo per favorire la riusabilità del codice. Invece di noiosi "copia-e-incolla", se abbiamo una serie di funzioni che ci interessano in una classe, basta importare tale classe nel nostro progetto per avere subito tutti i metodi di quell'oggetto.

Leonardo "Leo72" Miliani

PHP e MySQL

Come recuperare dei dati da un database MySQL con PHP

Salvare dei dati utilizzando i classici file di testo non sempre è conveniente, soprattutto se cominciano ad essere tanti e molto articolati.

Proviamo a farlo utilizzando il php, funzione che ci può venire molto utile anche con gambas, dato che nessun (o quasi) database su hosting professionali permette l'accesso da esterno (per farlo con gambas è possibile utilizzare il metodo get di http). Per prima cosa creiamo il form che eseguirà la nostra query. Questo è semplice HTML (a lato).

```
<html>
<head>
<title></title>
</head>
<body>
  <form name="frmLogin" action="query.php" method="POST">
  <h2>inserire nella casella il nome da cercare</h2>
  <input type="text" size="16" name="idutente"><br>
  <input type="submit" size="16" name="cmdsubmit">
  </form>
</body>
</html>
```

Spiegandolo molto velocemente, tramite il tag *form* inserisco un form, che come azione invierà (*post*) i dati alla pagina `query.php`. Il tag `input` servirà per creare una casella di testo (`text`) e un bottone per l'invio (*submit*). Ora passiamo alla pagina `query.php`, scritta in PHP (Listato 1).

Prima di analizzare il codice, vediamo come è costituito il nostro database. Esso avrà nome "nomi" e conterrà una tabella di nome "utenti" con alcuni campi fra cui "nome", che conterrà appunto il nome dell'utente da cercare.

Tramite il primo comando:

```
$casella = $_POST['idutente'];
```

prenderemo il dato precedentemente inviato e lo salveremo in quella variabile (se vi ricordate nella pagina HTML come attributo "name" alla casella di testo avevamo attribuito "idutente").

Ora passiamo ad assegnare i valori per la connessione:

```
$dbhost="localhost";
```

serve per scegliere l'*host* dove risiede il nostro database, solitamente è *localhost*.

```
$dbname="nomi";
```

serve per assegnare il nome al database, nel nostro caso il nostro DB si chiamerà "nomi".

```
$dbuser="root";
```

in questa variabile imposteremo il nome utente del database.

```
$dbpass="";
```

e in questa la password. Per connetterci al database usando i dati immessi in precedenza basterà usare il seguente comando:

```
$conn=mysql_connect($dbhost, $dbuser, $dbpass)
or die("impossibile connettersi al database mysql");
```

Tramite questo comando:

```
mysql_select_db($dbname, $conn)
```

selezioneremo il database su cui operare; ricordiamoci di passare come secondo parametro la connessione creata.

Ora scriviamo la nostra *query*:

```
$sql = "SELECT * FROM utenti" ." WHERE nome='$casella' " ;
```

con cui selezioniamo tutte le righe dove il nome è uguale a quello inserito. Ora eseguiamo la *query*:

```
$res =mysql_query($sql, $conn)
```

I parametri sono molto semplici: il primo contiene il codice SQL da eseguire, mentre il secondo la connessione.

Mentre con questo codice:

```
$num=mysql_num_rows($res);
```

troviamo il numero di occorrenze della nostra interrogazione.

E per finire con:

```
$info = mysql_fetch_array($res);
echo "ciao $info[1] ,  $info[2]";
```

estraggo nella variabile info (che sarà un *array*) le informazioni del risultato della mia *query*, in cui avrò in posizione 0 il primo campo, in posizione 1 il secondo e così via... nel mio caso visualizzerò il secondo e terzo campo.

Con questo semplice esempio siamo andati a selezionare dei dati da un database: cambiando la nostra *query* (la variabile $sql) potremo compiere operazioni di inserimento, aggiornamento e molte altre.

Alessio "ealmuno" Moretto

```
<html>
<head>
  <title>Presa dati da database</title>
</head>

<body>
<?php
//acquisisce valori form
$casella = $_POST['idutente'];

//variabili per connessione al database
$dbhost="localhost";
$dbname="nomi";
$dbuser="root";
$dbpass="";

//connessione al database
$conn=mysql_connect($dbhost, $dbuser, $dbpass)
or die("impossibile connettersi al server mysql");

//selezione del db su cui operare
mysql_select_db($dbname, $conn)
or die ("impossibile selezionare il database $dbname");

 //comando sql da eseguire
 $sql = "SELECT * FROM utenti" ." WHERE nome='$casella' " ;

 //esecuzione sql
 $res =mysql_query($sql, $conn)
  or die("Errore");

  //estrarre il numero di occorrenze
  $num=mysql_num_rows($res);
  echo "la tua query ha trovato $num records";

  if ($num==0){
   print "nessun record trovato";
  }
  else{
   for ($i=0; $i<$num; $x++){
     $info = mysql_fetch_array($res);
     echo "ciao $info[0] ,  $info[0] <br>";
   }
  }
//chiudiamo la connessione
mysql_close($conn);
?>
</body>
</html>
```

Listato 1

Su che cosa e come realizzare un programma

Il computer come intelligente strumento di sussidio alle attività umane

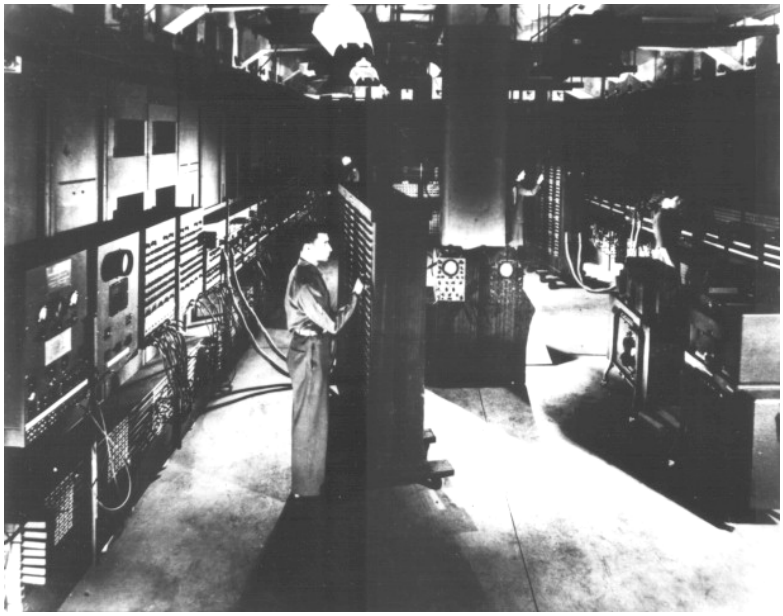
I primi approcci di programmazione avevano lo scopo di svolgere elaborazioni relativamente semplici, perché gli elaboratori non erano dotati di memoria e quindi le istruzioni utilizzabili potevano trattare pochi dati. Istruzioni e dati dovevano essere forniti in un formato comprensibile esclusivamente dall'elaboratore, e perciò chiamato formato macchina.

L'ideatore dell'allora nuova classe di elaboratori, successori dei precedenti sistemi meccanografici, fu un altro pioniere dell'informatica, l'ungherese John von Neumann. Egli ebbe l'ingegnosità di riprendere per il suo progetto la teoria del britannico Alan Mathison Turing, la cosiddetta "*macchina di Turing*" (o "automa di Turing"), formulata nel 1936 ed esistente esclusivamente come modello teorico.

Turing dimostrò nel suo studio che era possibile eseguire qualsiasi procedura di calcolo (algoritmo) per risolvere un problema in un numero finito di operazioni, dividendo il problema in una serie di problemi più semplici risolvibili quindi con operazioni semplici. Era l'idea del calcolatore universale. Nel suo studio rivolse la sua attenzione alla teoria di Boole, elaborata nel 1854 e conosciuta come *algebra booleana*. Essa concepisce procedure di calcolo effettuabili grazie ad operatori matematici (AND, OR, ...). Detti operatori, nella loro applicazione all'interno degli elaboratori, presero il nome di operatori logici e le operazioni che sono in grado di svolgere furono chiamate operazioni booleane. Le principali operazioni sono: AND, OR, NOT. Esse si basano sui simboli '1' e '0', dei quali '1' rappresenta la condizione di "vero" mentre '0' rappresenta la condizione di "falso".

Neumann concentrò perciò il suo studio sulla produzione di una macchina in grado di leggere, interpretare e modificare i valori '0' e '1', presenti in un determinato momento all'interno della macchina, anche se ancora non si parlava di memoria artificiale. Così, nel 1943, venne realizzato in Pennsylvania un enorme elaboratore, privo di parti meccaniche in movimento, l'**ENIAC** (*Electronic Numerical Integrator and Computer*).

Esso occupava un'intera stanza e pesava oltre 30 tonnellate. Per il trattamento dei dati era dotato di 17.000 valvole termoioniche. Ciascuna valvola poteva assumere lo stato di



acceso o spento, corrispondente ai valori '1' e '0' dell'algebra booleana. La macchina non era dotata di una vera e propria memoria: possedeva una pseudo-memoria, detta memoria a flip-flop; ogni unità di memoria era formata da due valvole che ad ogni eccitazione cambiavano stato. Complessivamente, l'ENIAC riusciva a memorizzare solo dieci parole, ma per farlo venivano impiegate migliaia di valvole.

Fu durante la costruzione dell'ENIAC che lo scienziato John Tykey coniò il termine *bit*, ricavandolo dall'unione abbreviata delle parole inglesi *binary* e *digit* (cifra binaria); esso rappresenta la più piccola unità di informazione relativa ai dati in trattamento nell'elaboratore ad un certo istante e specifica uno dei due stati che poteva assumere una valvola (per restare nell'ambito degli elaboratori di quel periodo), cioè acceso=1, oppure spento=0 . Ne consegue che l'elaboratore, sia esso uno di quei colossi di quegli anni, sia un avanzatissimo computer dei nostri giorni, possiede un suo alfabeto, l'alfabeto binario, da cui ha preso vita il linguaggio macchina, cioè il linguaggio dei computer. Ma allora, se il bit è l'unità elettronica più piccola rappresentata in un elaboratore, quanti bit occorrono per rappresentare un carattere comprensibile da un essere umano? Come le diverse combinazioni di un certo numero di lettere dell'alfabeto umano danno vita a parole comprensibili dall'uomo, così, la riunione combinata dei due elementi

dell'alfabeto binario dà forma a "parole" comprensibili dal computer e l'insieme di queste ultime forma le "frasi": una frase del linguaggio macchina può essere un gruppo di dati in trattamento, oppure un'istruzione in esecuzione; la "parola" richiede un ulteriore approfondimento.

Essendo la "parola" elettronica formata da un gruppo di bit, essa può appartenere perciò ad un sistema di di alfabetizzazione diversa, a seconda del numero di bit che compongono ciascuna di esse; perciò si dice che: una parola appartiene al sistema di alfabetizzazione ottale, chiamato semplicemente sistema ottale, se è formata da 6 bit consecutivi, mentre appartiene al sistema esadecimale, se è formata da 8 bit consecutivi. L'insieme di bit che formano una parola viene chiamato "*Byte*". Esso è l'unità di misura di memorizzazione composto, nel sistema di numerazione esadecimale, da 8 bit. Le combinazioni possibili dei bit che costituiscono il byte sono contenute nei due seguenti estremi: 00000000 ÷ 11111111, per un totale di 256 diverse combinazioni, corrispondenti a 2^8=256 possibili valori, da 0 a 255 (2^8-1=255->11111111_2). Per semplicità, qui si fa riferimento al solo sistema esadecimale. Esso è un sistema numerico in base 16, simboleggiato dai numeri 0÷9, seguiti dalle lettere A÷F. Il valore decimale 16 corrisponde al valore esadecimale '10'. La sua applicazione nel mondo degli elaboratori si deve essenzialmente alla sua stretta relazione col linguaggio macchina, basato sul sistema binario, ed alla capacità di rappresentare, in una sola notazione (8 bit), un numero di tre cifre.

Partendo da tale riscontro, dallo studio fatto nei laboratori di ricerca IBM, il byte è stato diviso logicamente in due sottoinsiemi di 4 bit ciascuno detti `Nibble`, cioè in due semibyte; essendo perciò ciascun semibyte formato da 4 bit, il valore decimale di ciascun semibyte è rappresentato dalla somma delle potenze di "2" dei bit "1" presenti nel semibyte; tenendo conto che il calcolo va effettuato partendo dal bit posto all'estrema destra del semibyte, si ha:

$1100=\Sigma(2^3*1 + 2^2*1 + 2^1*0 + 2^0*0)$='0C', cioè 12 del sistema decimale

Infatti:

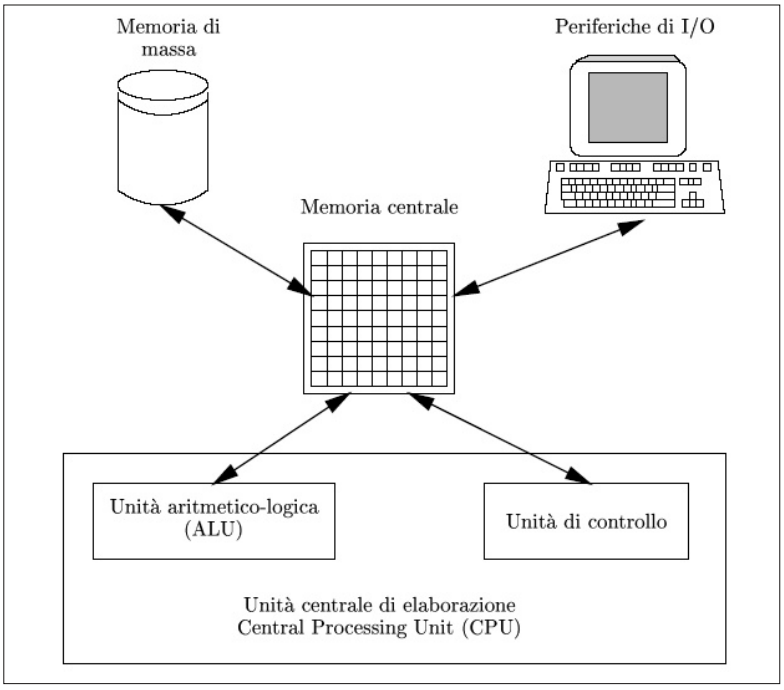
```
2³*1 = 8
2²*1 = 4
2¹*0 = 0
2⁰*0 = 0
      ===
      0C
```

Il sistema esadecimale fu applicato ai calcolatori elettronici per la prima volta dall'IBM nel 1963.

Qualcuno potrebbe notare: ma sono trascorsi 20 anni dalla nascita dell'ENIAC, il primo elaboratore più famoso degli anni '40. E nel frattempo come si è sviluppato il pianeta "computer"? L'ENIAC, nonstante la sua costituzione delicata, assolutamente sproporzionata rispetto alla sua mole, visse fino al 1955; però, già subito dopo la sua creazione, Neumann si applicò per aggiungere alla sua macchina la capacità di modificare il proprio comportamento, cioè di dotarla di una vera e propria memoria artificiale e, nel 1945, nacque l'`Edvac` (*Electronic Discrete Variables Automatic Computer*), la prima macchina digitale programmabile, basata su un'architettura ideata dallo stesso Neumann: per questo fu chiamata "`architettura di von Neumann`". Il programma, che fino a quel momento era costituito da poche istruzioni, capaci di svolgere poche operazioni solo nell'immediato, perché non poteva essere trattenuto a lungo dalla memoria a flip-flop, cominciava finalmente a marcare la sua impronta nella vita dei calcolatori. L'Edvac, a causa di dispute sulla paternità, entrò in funzione solamente nel 1951.

N.B.: tutte le immagini riportate sono state prelevate da it.wikipedia.org

L'architettura di von Neumann prevede un'unità elettronica di lavoro (CPU), nella quale risiedeva la memoria centrale con la



sua ALU (unità aritmetico-logica) e la sua unità di controllo, un'unità di input ed un'unità di output per l'entrata dei dati da elaborare e l'uscita di quelli già elaborati, nonché un canale (Bus) in grado di collegare tutti i componenti fra loro. Faceva la sua prima apparizione il nastro magnetico per la lettura/scrittura dei dati.

Il programma doveva essere caricato in memoria di volta in volta. La memoria di cui era dotato l'Edvac era chiamata a linea di ritardo.

Nel tempo immediatamente successivo la struttura della memoria elettronica subì notevoli progressi; la struttura a linea di ritardo fu soppiantata dalla memoria a

tamburo e, finalmente, negli anni '50 arrivò un tipo di memoria, nello stesso tempo affidabile, di piccole dimensioni ed a basso consumo: la `memoria a nuclei magnetici`. Quest'ultima fu sostituita, a partire dal 1970, dalla memoria a semiconduttori.

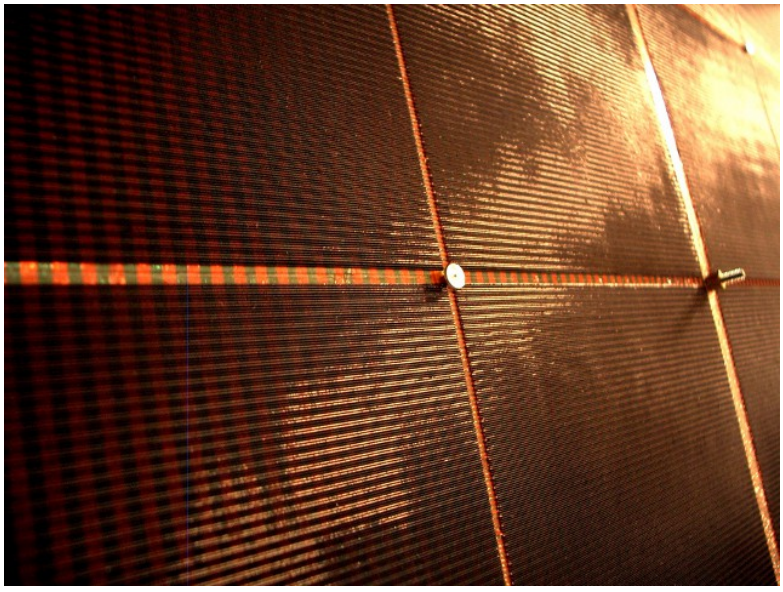
La memoria a nuclei magnetici merita una una particolare attenzione perché portò una piccola rivoluzione nella gestione dei computer: era una struttura cosiddetta non volatile, perché i banchi che la componevano erano costituiti da nuclei di ferrite, piccoli anelli magnetici disposti a griglia bidimensionale. Gli anelli erano percorsi da fili che servivano a portare i segnali di gestione della memoria. La memorizzazione dei dati avveniva per mezzo della variazione della polarità del loro campo magnetico. Ogni anello era così indirizzato in maniera assoluta e memorizzava un singolo stato digitale, determinato ad ogni cambio di polarità, per cui lo stato raggiunto con l'ultimo cambio di polarità rimaneva inalterato fino alla successiva variazione di polarità. Ciò permetteva, in caso di spegnimento dell'elaboratore di mantenere lo stato magnetico raggiunto subito prima dello spegnimento della macchina, ritrovandolo inalterato al successivo riavvio.

Impilando in modo opportuno un certo numero di griglie si poteva ottenere la lettura o scrittura di una parola. Si trattava di una struttura a castello (banco) organizzata in piani, ed occorrevano diversi banchi di memoria per ottenere grandi dimensioni. Le prime macchine prodotte possedevano una memoria di 1000 parole di 44 bit (in seguito portata a 1024 parole; in termini di oggi 5,5 Kbyte.)Il programma doveva essere caricato in memoria di volta in volta.

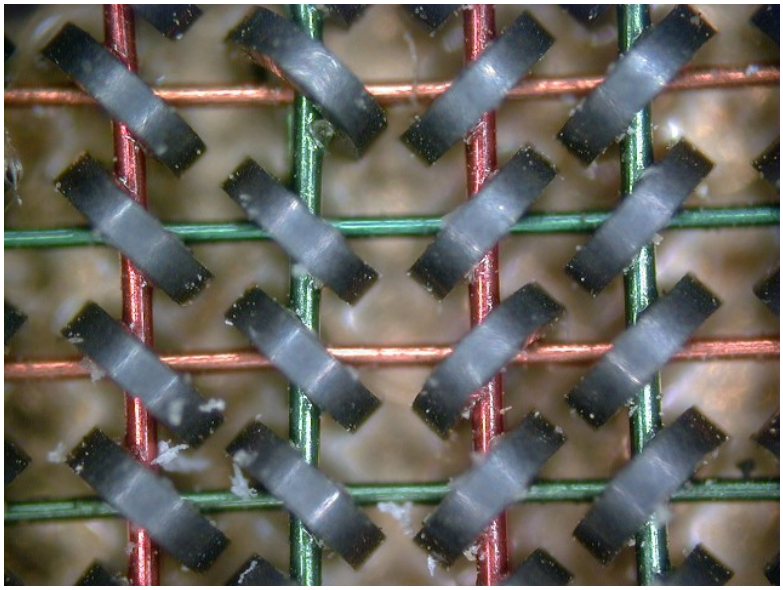
La grossa conquista permise di compiere passi da gigante anche alla programmazione che potè diventare prepotentemente una variabile fortemente cooperante con la struttura fisica della macchina, cioè con la sua parte dura (hardware).

(fine 3ª parte)

Pietro Antonino "Picavbg" Catania



Parte di 36 banchi di memoria a nuclei magnetici, residenti su scheda a doppia faccia, per un totale di 1,5 Megabit



Particolare di una memoria a nuclei magnetici: la distanza tra gli anelli è di circa 1 mm

Guida al C in puntate - Parte 2ª

Impariamo a programmare in C

Le variabili e le costanti in C

Per ***variabile*** si intende una porzione di memoria atta a contenere un determinato dato che potrà poi essere modificato nel corso dell'uso del programma. Essa può essere inizializzata all'atto della sua dichiarazione con un valore di *default* o può essere dichiarata senza alcun valore. In tal caso il linguaggio non garantisce un valore iniziale utile. Ad esempio una variabile può essere dichiarata ed inizializzata in due modi diversi.

Per doppio passaggio:

```
// dichiaro
int x;
// inizializzo
x = 14;
```

o direttamente:

```
// dichiaro ed inizializzo
int x = 14;
```

Una ***costante***, invece, è una porzione di memoria che viene dichiarata per contenere un determinato tipo di dato e ne conserva il valore per tutta la durata dell'uso del programma senza mai poterne modificare il valore.

Ogni variabile o costante nel C deve essere identificata da un nome al quale si farà poi riferimento nel corso dello svolgimento del programma per leggere o scriverci dati. Il nome delle variabili in C è arbitrario e può avere lunghezza variabile però il compilatore si interesserà solo delle prime 8 lettere che lo compongono. Quindi anche se a prima vista un dato di nome "variabile1" par diverso da "variabile2" il compilatore restituirà errore poichè vedrà due variabili con lo stesso nome ovvero "variabil"

Il C è un linguaggio che basa la sua struttura sui dati. Dati diversi, infatti, equivalgono a cose diverse. Questo tipo di gestione permette di avere un buon uso dela memoria perché ad ogni tipo di dato è dedicata la giusta quantità di memoria e in più dà la possibilità di usare o creare strutture *ad hoc* per le diverse esigenze di sviluppo.

Il C gode delle presenza di 6 tipi di dati base:

```
char
int
long
short
float
double
```

Il `char` è il dato adibito alla gestione dei caratteri. Ogni volta che viene dichiarata una variabile di tipo *char* essa occupa una quantità di memoria pari a 8 bit (1 byte). Il *char*, nonostante sia effettivamente un tipo di dato numerico, permette la gestione del suo valore come una normale "lettera". Un *array* (vettore) di *char* permette la virtualizzazione di una stringa mentre, in altri linguaggi, esiste ormai un tipo di dato *string* dedicato. Un esempio di dichiarazione di un dato *char* inizializzato è:

```
char carattare = 'a';
```

E' importante notare come definire il carattere '8', ad esempio, è totalmente diverso dalla definizione di un valore intero 8. Un *char* che contiene il carattere '8' in realtà avrà al suo intero il valore numerico che, in base alla codifica in uso, equivale a quel determinato carattere mentre una variabile numerica che contiene 8 avrà proprio quel valore preciso in memoria.

L'*int* è il tipo adibito a contenere dati numerici di numeri interi o meglio definiti come numeri naturali. Tali numeri, pertanto, devono contenere numeri senza virgole, frazioni, parti immaginarie, etc... un dato di tipo *int*, all'atto della sua dichiarazione, occupa 16 bit (2 byte). I valori che un *int* può assumere vanno da -32.768 a +32.767 se si usa il segno (valori positivi o negativi) o va da 0 a 65535 per soli numeri interi. Un esempio di dichiarazione di un dato *int* inizializzato è:

```
int numIntero = 14;
```

Il `long` può essere visto come un particolare *int* creato appositamente per ovviare al problema di uso di numeri più grandi di quelli che l'*int* offre. Esso occupa una quantità di memoria paria a 32 bit (4 byte) ed offre la possibilità di usare valori che vanno da -2.147.483.648 a +2.147.483.647 per numeri con segno o da 0 a 4.294.967.295 per numeri privi del segno.

Un esempio di dichiarazione di un dato *long* inizializzato è:

```
long numIntLong = 1000000000;
```

Lo `short` è un dato molto simile all'*int* ma

Ogni variabile o costante è definita da un nome. Tale nome deve essere, per buona norma, mnemonico, ovvero devre ricordare a cosa serve il dato. Non possono esserci due variabili o costanti con lo stesso nome

che trova limitazioni più restrittive per quanto riguarda il suo valore massimo o minimo. Esso infatti occupa 8 bit (1 byte) e può perciò coprire un *range* che va da -127 a +128 pe numeri con segno mentre va da 0 a 256 per numeri senza segno.

I `float` e i `double` sono particolari tipi di dati che danno la posibilità di rappresentare i numeri in virgola mobile, ovvero i numeri reali. La differenza tra i due tipi di dati sta nella quantità di memoria utilizzata e pertanto nell'accuratezza con cui i numeri possono essere rappresentati. Un *float*, infatti, richiede una quantità di memoria pari a 32 bit (4 byte) mentre un double richiede 64 bit (8 byte). Un esempio di dichiarazione di un dato *float* e di un *double* inizializzato è:

```
double dblVirgola = 5.023;
float fltVirgola = 7.8;
```

SIGNED ed UNSIGNED

Come si è visto, ogni dato ha la possibilità di considerare l'insieme dei suoi valori limitato diversamente in base all'uso che se ne vuole fare. Ad esempio due dati, entrambi *int*, potranno assumere valori massimi diversi se uno è sempre positivo mentre l'altro assume sia valori positivi che negativi. E' importante avere in mente tale concetto per poter avere a disposizione sempre il miglior tipo di dato per le nostre esigenze. Di default i dati in C vengon visti come *signed* e perciò è possibile usare il segno per definire il valore in maniera positiva o negativa. Qualora ci fosse bisogno di definire esplicitamente che un numero non necessita di una rappresentazione negativa allora si può dichiarare la viariabile in questo modo:

```
unsigned int Numero = 15;
```

altrimenti è possibile usare la sintassi:

```
signed int Numero = -15;
```

che è equivalente a

```
int Numero = -15;
```

COSTANTI

Il discorso trattato fin ora era studiato per l'uso delle variabili; tuttavia, tutte le regole di cui sopra valgono anche per quanto riguarda il trattamente di dati costanti. Bisogna comunque ricordare che essi mantengono il valore iniziale

e perciò quanto detto vale solo all'atto della dichiarazione e mai durante lo svolgimento del programma. Un esempio di dichiarazione di una costante può essere:

```
const float PIGRECO = 3.14;
```

Da notare la presenza della parola chiave *const* che definisce il dato come costante.

CASTING E CONCLUSIONE

Visto il tipo di gestione dei dati in C, esso viene definito come linguaggio a tipizzazione statica, ovvero dove i tipi di dati vengono definiti all'inizio e mantenuti pressoché costanti per la durata del software. Tuttavia esiste una pratica definita `casting` mediante la quale è possibile cambiare il tipo di un dato per poterlo adattare ad una certa circostanza. Il *casting* si avvale della sintassi del tipo:

```
variabile1 = (tipo)variabile2
```

Ad esempio per ottenere un *int* da un valore *float* è possibile seguire il seguente codice di esempio:

```
float fltNumero = 2.36;
int intNumero;

intNumero = (int)fltNumero;
```

In questo modo intNumero assumerà valore 2, ovvero la parte intera del numero *float*.

Conoscere i dati in C significa essere a conoscenza delle fondamenta di un linguaggio ed è perciò importante studiare bene l'argomento prima di addentrarsi in argomenti più avanzati.

Francesco "Ceskho OpenCode" Apruzzese

Nel C il nome delle costanti e delle variabili e case sensitivi. Ovvero due nomi uguali ma che presentano lettere maiuscole e minuscole differenti vengono viste come due dati diversi, Pertanto "Numero" sarà un dato diverso da "NUMERO"

E' buona norma di programmazione definire i nomi delle variabili in lettere minuscole e quelli delle costanti in maiuscolo

GestHotel

Gestionale alberghiero Open-source

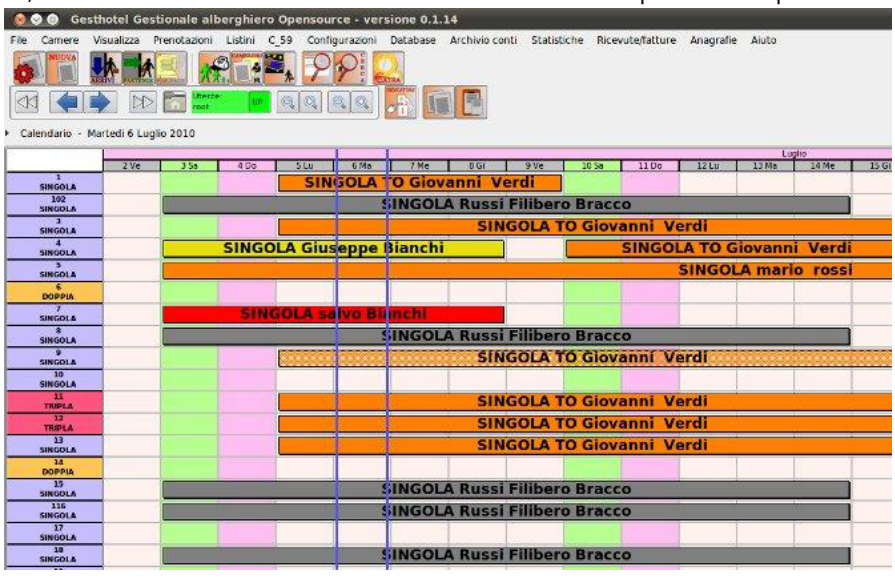
In campo turistico sono pochi i software che contano di funzionalità complete a livello di gestione. Tanti sono i software ma pochi sono quelli che permettono di ottimizzare il lavoro contenendo i costi. Infatti chi gestisce un albergo sa a priori che una buona parte delle spese iniziali riguarda anche la scelta di un programma gestionale che permetta di gestire al meglio tutte le varie fasi del soggiorno clienti, dalla prenotazione fino a terminare con il *check-out*.

Da gennaio 2010 *Fea Sergio* (programmatore) e *Gaetano Lo Nigro* (grafico) hanno iniziato la creazione di un gestionale per il settore alberghiero, grazie anche a consigli, idee e suggerimenti di molti utenti GNU/LINUX: naturalmente il software è *open-source* e distribuito sotto licenza GPL3.

Il programma si chiama **GESTHOTEL**, è scritto in ***Gambas2*** e, nonostante la giovane età, promette di essere un software molto interessante per gli addetti al settore sia per la semplicità di utilizzo sia per la versatilità.

Dotato di un'interfaccia grafica intuitiva, permette la completa gestione delle operazioni necessarie in molteplici strutture ricettive che possono spaziare dal B&B all'hotel passando anche per il campeggio.

Cuore del programma è il *planning* grafico, tramite il quale è possibile effettuare le principali operazioni di prenotazione, *check-in*, *check-out*.



La grafica completamente modificabile nelle colorazioni permette una veloce individuazione dell'occupazione delle camere.

Basato su un database MySQL, Gesthotel permette l'utilizzo su più postazioni connesse in rete oppure su singola postazione. Inoltre è possibile abilitare la gestione utenti fino a tre livelli: ROOT, SLAVE, READ-ONLY. Ci troveremo quindi in una situazione in cui solo l'utente ROOT potrà effettuare le modifiche alle impostazioni dei listini, delle camere ecc., mentre l'utente SLAVE potrà effettuare le normali operazioni di gestione alberghiera; l'utente READ-ONLY potrà invece solo visionare ma non effettuare nessuna operazione di modifica.

Il programma, in fase di sviluppo (attualmente nella versione 0.1.13), è già utilizzato in alcune installazioni alberghiere: le frequenti innovazioni e implementazioni non influenzano eventuali dati inseriti nel programma, ma arricchiscono Gesthotel di funzionalità.

Esiste inoltre un'utile funzione di *backup* del database impostabile per un auto-backup giornaliero che permette di ripristinare la funzionalità del programma e la salvaguardia dei dati anche per le installazioni non di testing.

Gesthotel permette di essere utilizzato sia da strutture che utilizzano i listini periodici con conteggio sia a persona che a camera, sia da strutture che utilizzano dei portali per definire il costo di una camera giorno per giorno; inoltre è possibile impostare anche dei pacchetti speciali ad esempio per *week-end* o altro.

Naturalmente vengono gestiti oltre alle camere anche altri servizi "accessori" (es: garage, sauna, ecc.) nonché l' addebito di extra (bar, servizio in camera, ecc).

Alcune delle principali funzioni sono:

- prenotazione a listini (numero di listini e servizi infinito)

- prenotazioni best rate

- prenotazioni per pacchetti speciali

- gestione servizi accessori

- gestione extra

- stampa schede PS

- stampa moduli C59


Mod. ISTAT C/59

ISTAT

Num. Progr.

ISTITUTO NAZIONALE DI STATISTICA

ENTE NAZIONALE ITALIANO PER IL TURISMO

RILEVAZIONE DEL MOVIMENTO DEI CLIENTI NEGLI ESERCIZI RICETTIVI

Anno Mese Giorno

Comune

Tipo di esercizio

(Albergo, campeggio, ecc.)

Denominazione

Numero stelle

Clienti del giorno precedente

Arrivati

Totale

Partiti

Presenti nella notte

CLIENTI DEL GIORNO

- stampa conti in carta semplice, fattura, ricevuta, proforma

- stampa movimentazione camere

- stampa foglio cameriere

- gestione overbooking

- stampa presenze

- stampa statistiche occupazione camere

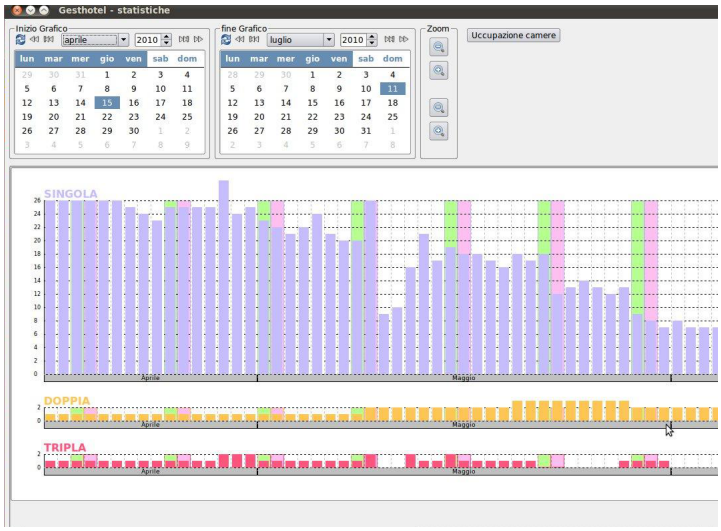
- gestione sconti per agenzie o clienti

- anagrafica clienti

- tanto altro ancora

È stata posta tanta attenzione nella cura dei particolari del programma specie nella creazione dei vari form che garantiscono un semplice ed immediato impatto, al fine di favorire le varie operazioni alberghiere nella maniera più semplice ma completa possibile.

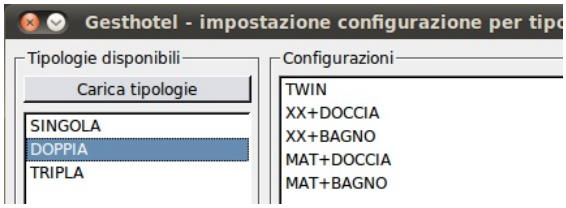
La stampa della documentazione necessaria ai fini legislativi e l'implementazione dell'anagrafica clienti (con possibilità di invio email) rendono il software usufruibile anche nelle strutture con uno svariato numero di camere.



Un sistema di gestione delle fatture/ricevute con possibilità di effettuare anche delle dirette (non legate a prenotazioni) sono uno dei punti di forza del programma.

Ultimamente è stato implementato anche un sistema di statistiche che permette di tenere sotto controllo la situazione delle camere presenti nella struttura in cui si andrà ad operare, nonchè un sistema di controllo camere in percentuale, ottimo per chi si occupa di *Revenue Management* e vuole sempre avere una rapida consultazione dei dati Occupazione.

Per quanto riguarda la gestione delle camere, Gesthotel permette una completa personalizzazione sia del numero delle stanze sia della loro tipologia: questa funzione è pensata appositamente per le strutture in cui si richiede una diversa organizzazione dei posti letto (singola, doppia, francese ecc.).



Gesthotel, nelle ultime versioni, permette inoltre la creazione di un file utilizzabile per l'invio telematico delle schede di pubblica sicurezza, tramite il servizio "alloggiati web" ormai attivo in tantissime province italiane. Tramite questa utility il risparmio di tempo impiegato nella consegna è notevole, ed inoltre non si correrà più il riscio di consegne ritardatarie.

Naturalmente il software non è ancora alla sua versione definitiva, e ognuno può contribuire con opinioni, suggerimenti o idee all'ampliamento del programma tramite un apposito forum dove si può peraltro trovare supporto in caso di necessità :

`http://www.tutto-opensource.org/gesthotel/smf/index.php`.

Gesthotel si propone come il primo Gestionale Alberghiero Opensource adatto a diversi tipi di strutture ricettive: la forza di questo software sta nella rivoluzione che potrebbe portare nel campo turistico proponendo un'alternativa valida ai più diretti concorrenti, molto più blasonati, abbassando notevolmente i costi di gestione.

Consiglio: Gesthotel è installabile su tutte le principali distribuzioni Linux, tuttavia su alcune distribuzioni un componente Gambas non viene installato correttamente: in quel caso suggerisco di dare da terminale

`sudo apt-get install gambas2-gb-db-mysql`.

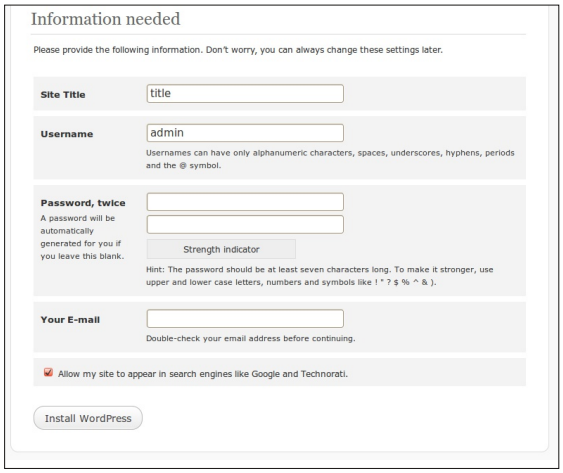
Fea "fsurfing" Sergio Gaetano "tangoku" Lo Nigro

WordPress 3.0

Tutte le novità

WordPress, che è forse la piattaforma più utilizzata nel web per la gestione di blog, è ormai giunta alla versione 3.0, che prevede importanti novità.

La prima la si può notare subito in fase di installazione: infatti sarà possibile inserire il nome utente dell'amministratore che, quindi, non sarà più *admin*, come lo era di default nella precedente versione.



Dal punto di vista dei post ci sono due novità (nel tema predefinito), uno nella galleria di immagini e uno nelle news: infatti aggiungendo come categoria "gallery" ad un post esso sarà visto in maniera differente da un normale post, mostrando solo un'immagine con a fianco il numero di immagini di quella galleria.

Altra novità è la categoria "asides" che permette di mostrare delle news, a cui viene eliminato il titolo nella loro visualizzazione in homepage.

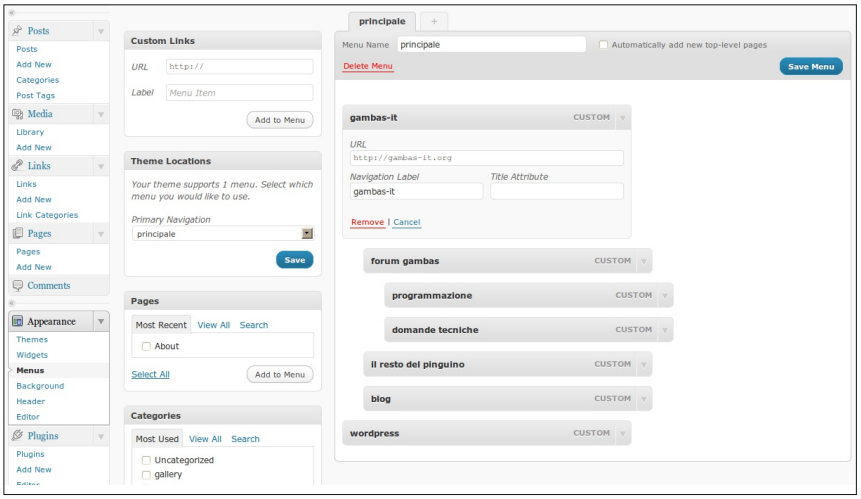
Un altro cambiamento è nell'esportazione del nostro blog: andando infatti in *Tools-->Export* ci verrà data la possibilità di esportare il blog con i post che hanno solo alcune caratteristiche, tipo quelli compresi tra due date, quelli di un certo autore e quelli con una certa categoria.

Altra importante novità è la possibilità di avere un `multiblog`, cioè gestire più blog WordPress con una sola installazione: questo era già possibile con Wordpressmu, ma da questa nuova versione è stato integrato tutto assieme. Per attivare questa funzionalità basta inserire alla fine del file *wp-config.php* questa riga di codice:

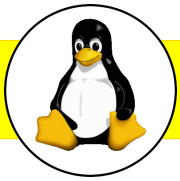
```
define ('WP_ALLOW_MULTISITE', true);
```

Andando poi nella sezione *Tools-->Network* sarà possibile configurare il tutto. Oltre a tutte le modifiche riguardanti la gestione dell'aspetto (sempre parlando del tema di default), quali la possibilità di inserire uno sfondo personalizzato, di modificare l'*header* ed i *widgets* è possibile da questa versione modificare il menu. Questa innovazione, forse un po' in ritardo a mio parere, è la più importante per questa versione, perché rende WordPress un CMS a tutti gli effetti.

Sarà infatti possibile creare un menu con un semplice click del mouse, assegnandogli come destinazione una pagina web, una categoria, un post o molto altro.



Alessio "ealmuno" Moretto



La cattedrale ed il bazar

Due stili di sviluppo del software a confronto: il modello "cattedrale" e quello "bazar"

3. L'importanza di avere utenti

E così ho ereditato *popclient*. Fatto parimenti importante, ho ereditato gli utenti di *popclient*. Importante non soltanto perché la loro esistenza testimonia che stai rispondendo a un loro bisogno, che hai fatto qualcosa di buono. Coltivati in maniera appropriata, gli utenti possono trasformarsi in co-sviluppatori.

Altro punto di forza della tradizione Unix, portato felicemente agli estremi da Linux, è che molti utenti sono essi stessi degli hacker. Ed essendo i sorgenti disponibili a tutti, posso diventare degli hacker molto efficaci. Qualcosa di tremendamente utile per ridurre il tempo necessario al debugging. Con un po' d'incoraggiamento, ogni utente è in grado di diagnosticare problemi, suggerire soluzioni, aiutare a migliorare il codice in maniera impensabile per una persona sola.

6. Trattare gli utenti come co-sviluppatori è la strada migliore per ottenere rapidi miglioramenti del codice e debugging efficace.

È facile sottovalutare la potenza di un simile effetto. In realtà un po' tutti noi del mondo open source eravamo soliti sottovalutare drasticamente il fatto che tale potenza crescesse di pari passo con il numero degli utenti e con la complessità del sistema. Finché Linus Torvalds ci ha mostrato le cose in maniera diversa.

In realtà ritengo che la mossa più scaltra e consequenziale di Linus non sia stata la costruzione del kernel di Linux in sé, bensì la sua invenzione del modello di sviluppo di Linux. Quando ho espresso questo mio pensiero in sua presenza, sorridendo ha ripetuto con calma quel che va spesso affermando: "*Praticamente sono una persona molto pigra cui piace prendersi il merito di quel che sono gli altri a fare.*" Pigro come una volpe. Oppure, come avrebbe detto Robert Heinlein, troppo pigro per fallire.

Guardando all'indietro, un precedente per i metodi e il successo di Linux può esser trovato nello sviluppo della libreria Lisp GNU e per gli archivi del codice Lisp di Emacs. In opposizione allo stile di costruzione a cattedrale del nucleo centrale in C di Emacs e di gran parte di altri strumenti della Free Software Foundation (FSF), l'evoluzione del codice Lisp risultò assai fluida e guidata dagli utenti. Idee e prototipi vennero spesso riscritti tre o quattro volte prima di raggiungere una forma stabile e definitiva. E le collaborazioni estemporanee tra due o più persone consentite da Internet, alla Linux, erano evento frequente.

Non a caso il mio caso di "hack" precedente a *fetchmail* fu probabilmente la modalità Emacs VC, una collaborazione via email secondo lo stile Linux con altre tre persone, soltanto una delle quali (Richard Stallman, autore di Emacs e fondatore della FSF, `http://www.fsf.org`) ho poi avuto occasione di incontrare dal vivo. Si trattava di realizzare il frontale per SCCS, RCS e più tardi CVS dall'interno di Emacs, dotato di opzioni a "one-touch" per le operazioni di controllo. Ciò ha preso avvio da un minuto, crudo sccs.el scritto da qualcun altro. E lo sviluppo di VC ha avuto successo perché, al contrario dello stesso Emacs, il codice di Lisp riuscì a passare molto rapidamente tra diverse generazioni di distribuzione/test/miglioramenti.

(Continua...)

L'autore, *Eric Steven Raymond*, è un informatico statunitense nato nel 1957. E' lo sviluppatore del software **fetchmail**, è l'attuale manutentore del **Jargon File** (meglio noto come "Il nuovo dizionario degli hacker") e l'autore del **Glider**, il simbolo identificativo degli hacker. E' una figura prominente del panorama hacker internazionale e nel 1997 ha pubblicato un'analisi su due stili differenti di sviluppo del software libero. Il saggio, intitolato "**La cattedrale e il bazaar**", è generalmente considerato il manifesto del movimento open source.

Il resto del Pinguino pubblicherà il saggio a dispense per motivi di lunghezza dello stesso.

Il saggio è liberamente consultabile su WikiSource all'indirizzo: `http://it.wikisource.org/wiki/La_cattedrale_e_il_bazaar`

Gli autori

Elenco degli autori che hanno contribuito alla stesura degli articoli pubblicati su questo numero

Leonardo "leo72" Miliani

Mastico informatica fin da quando avevo 12 anni, essendo uno di quei fortunati che hanno conosciuto i mitici computer ad 8 bit degli anni '80 (il mio primo home computer è stato un Commodore 16). Da sempre patito per la programmazione, ho scritto codice in qualsiasi linguaggio. Da diversi anni utilizzo esclusivamente software libero perché la conoscenza è come l'aria: non se ne può fare a meno, non la si può negare a nessuno.

Alessio "ealmuno" Moretto

Frequento l'ITIS Informatica, amo i computer e tutto quello che c'entra con la tecnologia e il suo sviluppo, in particolare la programmazione e un po' di grafica.

Pietro Antonino "picavbg" Catania

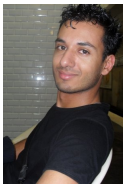
Sin da bambino ho avuto attrazione per le macchine calcolatrici e poi, più grandicello, per l'informatica. Diplomato in Ragioneria ho potuto, dopo alcuni anni, affacciarmi nel mondo del lavoro e lì, alla prima occasione, ho partecipato ad un test attitudinale di informatica che ho superato; da quel momento, ottenuta la qualifica di programmatore, ho potuto rapportarmi per quasi tutta la vita lavorativa con gli elaboratori elettronici e coi computer.

Francesco "Ceskho OpenCode" Apruzzese

Appassionato di tutto ciò che ha un processore. Ama programmare in qualsiasi linguaggio esista. Troppo intelligente per frequentare l'università. Il suo sogno è hackerare il proprio computer senza che lui stesso se ne accorga.

Fea "fsurfing" Sergio

Sono un elettrotecnico , appassionato di programmazione in gambas ed uno dei moderatori del forum gambas-it. Grato al mondo opensource per tutto quello che mi ha dato cerco di ricambiare nelle mie possibilità con piccoli programmi di utilità e divertimento.



Gaetano "tangoku" Lo Nigro

Sono un segretario d'albergo con la passione per l'informatica. Lavoro all'estero come receptionist/night auditor. Mi piace scrivere e fare piccoli lavoretti grafici utilizzando solo software opensource. Mi piace stare a contatto con le persone e amo tenermi informato su tutto quello che succede nel mondo.

Per contattare uno degli autori in merito ad un suo articolo (spiegazioni, approdonfimenti, correzioni, ecc...) potete scrivere a `ilrestodelpinguino@gambas-it.org` specificando l'articolo in questione ed il numero su cui è comparso il pezzo.

"Il resto del Pinguino" è la rivista digitale ufficiale di **Gambas-it.org**, la comunità italiana dei programmatori Gambas.
Pubblicazione aperiodica