PARTE 5

Il processo di programmazione

Problemi e algoritmi

- La **programmazione** e' la disciplina che si occupa della risoluzione di problemi tramite programmi scritti in qualche **linguaggio di programmazione**
- Il **problema** e' una definizione sintetica di cio' che il programma deve realizzare dal punto di vista dell'utente
- La **soluzione** fornisce un'idea dell'approccio che deve essere seguito per risolvere il problema, tenendo anche in considerazione quale software gia' esistente potrebbe essere riutilizzato
- L'**algoritmo** e' una descrizione precisa e non ambigua di una sequenza di passi da seguire per risolvere un problema. L'algoritmo e' quindi un raffinamento della soluzione. Gli algoritmi possono essere espressi in molti modi
- Il **programma** e' una (possibile) descrizione dell'algoritmo espressa usando un particolare linguaggio di programmazione. Si intende che il programma debba essere completo ed eseguibile
- Per i banali problemi che vedremo, useremo direttamente il linguaggio C per descrivere gli algoritmi. Quindi, per noi, **algoritmi=programmi**

Il processo di programmazione

- Il **processo di programmazione** comprende tutte le attivita' necessarie per sviluppare dei programmi in modo che siano memorizzati e preparati per l'esecuzione
- Per sviluppare programmi su un computer e' necessario un ambiente di sviluppo che almeno comprenda, oltre al sistema operativo, un **editor** ed un **compilatore**

Editor e compilatori

- Un **programma** e' un testo che consiste in una sequenza di istruzioni scritte in un particolare linguaggio di programmazione. Tale "testo" viene creato e salvato come un file su disco tramite un **editor**.
- Una volta che e' stato scritto e memorizzato, il (testo del) programma viene dato in input al **compilatore**
- Il compilatore ha una duplice funzione:
  - Controlla la **validita' del programma**: segnala gli **errori di sintassi**. Questi errori devono essere corretti tramite l'editor ed il programma corretto deve quindi essere nuovamente compilato
  - Se il programma non contiene errori, il compilatore **traduce il programma** in linguaggio macchina

Esecuzione dei programmi

- Il compilatore naturalmente non e' in grado di scoprire gli **errori logici o di esecuzione** (i ben noti **bug**) del programma: in tali casi, il programma risulta essere sintatticamente corretto ma non si comporta come ci si aspetta
- Il programma viene **eseguito** (o "**fatto girare**") attraverso un comando al sistema operativo che lo carica in memoria e quindi diventa un processo in esecuzione
- Gli errori logici possono essere prevenuti seguendo delle **buone regole di programmazione**
- Spesso gli errori logici si individuano mandando in esecuzione il programma ed osservandone il comportamento tramite degli **insiemi di test affidabili**
- Una volta rilevato e corretto un errore logico, si dovrebbero nuovamente eseguire i test sul suo comportamento, in quanto le presunte "correzioni" potrebbero aver introdotto nuovi errori
- Il processo di rilevazione e correzione degli errori logici di un programma viene chiamato **debugging**

HelloWorld in C

Nome file: helloword.c

```
#include<stdio.h>

main() {
  printf("Hello World!\n");
}
```

gcc helloword.c -o helloword.exe

Token

I token sono unita' sintattiche di base del C

- Parole chiave
- Identificatori
- Costanti
- Costanti stringa
- Operatori
- Simboli di interpunzione

Parole chiave (Keyword) del C

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

La libreria standard

Input e Output	<stdio.h>
Test dei caratteri	<ctype.h>
Funzioni su stringhe	<string.h>
Funzioni Matematiche	<math.h>
Funzioni Utilita'	<stdlib.h>
Funzioni Diagnostiche	<assert.h>
Liste argiomenti variabile	<stdarg.h>
Salti non locali	<setjmp.h>
Segnali	<signal.h>
Funzioni Date e Ore	<time.h>
Limiti dei tipi di variabile	<limits.h>

Tipi di Dato

Un singolo byte	char
Numero Intero	int
Numero Intero corto	short
Numero Intero lungo	long int
Numero in virgola mobile (singola precisione)	float
Numero in virgola mobile (doppia precisione)	double
Numero in virgola mobile (precisione estesa)	long double
Interi con segno	signed {char/int}
Interi senza segno	unsigned {char/int}