Prefazione

Salve parliamo di programmazione in C per ora, poi si tratterrà il C++ , prima di iniziare è bene sapere che io sono solo una persona che ho sempre studiato da autodidatta iniziai con le scuole basse si con l' elementari, certamente trovando moltissimi problemi il motivo che sono stato costretto ad essere un' autodidatta è il fatto che stavo sempre male , cosi alla scuola andavo quando non andavo per le varie visite mediche, tutto questo è durato fino i 17 anni pertanto la mia scuola è bassa e da autodidatta in tutto , sono stato bocciato anche parecchie volte ma non era importante ho lavorato come autista pendevo una bella paga, molto più di uno che risponde al telefono nonostante sia laureato, troverete dell' errori in questi miei messaggi ma non importa , se ero uno scienziato non avevo bisogno di capire l' eccelso mondo della scuola

Leonardo da Vinci possiamo esserlo tutti nel nostro piccolo basta dire voglio essere bravo per fare questo e dai nostri errori si impara meglio , per Leonardo c'era un comune denominatore lavorare la pietra il ferro il legno basta capire che la sostanza è sempre uguale fatta dalla stessa materia, l' importante la nuova idea che porta i nuovi traguardi

Imparare vuol dire solo avere esperienze , le vere esperienze non nascono sopra un libro ma da quello che facciamo e da quello che siamo capaci di comprendere, bello insegnare le sequenze di Fibonacci ma in vita cosa serve ? poi non sapete che cosa rappresenta il numero 666 della bestia di cui i miei libri, scritti dai miei servi parlano, Apocalisse 13, non rappresenta Satana ma la statua cui parla il mio servo Daniele 3, la statua che misura 6x6x60 cubiti che rappresenta dove tutti voi vi inginocchiate per essere adoratori d'idoli e questo vi potrà salvare la vita cosi potrete entrare nella mia Nuova Gerusalemme

Milioni di persone tagliano un albero alto 30\60 metri lo fanno cadere preciso dove vuole nessuna scuola o libro ti insegna questo

Le cose da quando è mondo si tramandano da padre in figlio quante persone hanno fatto cose impensabili per altri, vedete persone che facevano salumi dei propri animali di casa creare insaccati e tramandarli di padre in figlio oppure creare il formaggio dal latte queste persone hanno studiato ? eppure i salumi e i formaggi si fanno da quando si è creato l' uomo infatti già i miei figli Adamo e Eva sapevano cosa fare e come farlo, già loro usavano il fuoco per mangiare le cose cotte i loro figli Caino e Abele Genesi 4 erano già contadini uno per il terreno l' altro per l' animali cosa ci faceva con il latte Abele ? siete voi che portate storie strampalate con le vostre ricerche, il dinosauro o il drago era uguale come un iguana oggi e aiutava l' uomo

Vedete i ponti Romani creati dai soldati ancora dopo millenni sono li , i nostri ponti creati dai nostri bravi laureati il giorno dopo cadono perché ?

01) Preprocessore o precompilatore le sue key o comandi

Prima di iniziare il vero cammino dobbiamo conoscer come iniziare la prima cosa che dobbiamo inserire nei programmi in C sono i comandi meglio direttive del preprocessore cosa è il preprocessore ? sembra che sia una macchina dentro il PC , no sono dei comandi prestabiliti che fa parte del compilatore , infatti il preprocessore è meglio chiamarlo precompilatore inizializza i comandi che dobbiamo inserire se già avete letto la guida di webMaster che avevo consigliato : http://www.mrwebmaster.it/c/primo-programma_9691.html

Avete visto che per iniziare si deve inserire la formula magica

"#include <libreria> "

Quella libreria che era STDIO.H è il comando che fa riconoscere il significato della parola printf che dice al compilatore scrivi su monitor ogni libreria fa riconoscere i suoi comandi il significato di stdio proprio il IO sta per INPUT entrata OUTPUT uscita un input è la tastiera, il mouse , una telecamera , l' output è il video , il disco fisso, la stampante ecc...
Questi comandi per pilotare il compilatore in C sono proprio pochi :

#define #error #include #elif #if #line #else #ifdef #pragma #endif #ifndef #undef #warning

Quelli più importanti sono #INCLUDE e #DEFINE il primo serve per includere i vari file che pilota il compilatore tutti i file li troviamo nella cartella include del compilatore ma sono tantissimi perché qui nel VS2015 c'è la libreria del C , del C++, del C++ MFC, del C++ CLR, C ATL, per questo bisogna impostare il compilatore solo in C altrimenti da errori potevamo usare altri compilatori ma questo è molto più completo poi si può lavorare con il visuale dopo capito tutto l'altro

Inoltre con #INCLUDE " file.* " e virgolette possiamo aprire i file esterni

quando il compilatore trova questo comando #Include sa che deve leggere dentro quel file e mette in memoria tutti i comandi che serve per gestire la compilazione , infatti se togliete l' include inserendo 2 // per i commenti noterete che i comandi collegati come printf e scanf saranno sottolineati in rosso e se mandi in esecuzione ti dice nella finestra errori output

warning C4013: 'printf' non definita

Se ancora funziona normale è perché il file che hai aperto aveva gia caricato i comandi di STDIO, #DEFINE serve per definire delle parole come se fosse delle costanti una variabile fissa che il programma non può più cambiare esempio dobbiamo impostare una linea anzi che scrivere sempre ;

printf("--");

possiamo fare

#define li printf(" --");

cosi ogni volta che si richiama li il programma avviato stampa la linea ora vediamo questi che si usano con #define come non lo capito bene ovvero ho capito che funziona sulle macro come se fosse if dovrebbe togliere le definizioni di #Define per poi sostituirle .... anche qui si usa i comandi di #IF

#IFDEF e #IFNDEF #UNDEF le parole significa : " se definito ", " se non definito", " elimina la definizione "

Se si usa #Undef il compilatore non legge più la definizione pertanto non troviamo più quello che era stato dichiarato con #define alla fine dell' uso di #Ifdef va sempre chiuso con #endif altrimenti il compilatore da errori

come funziona veramente non lo capito, se trovate qualche testo che spiega bene potete segnalarmelo

Le direttive #if #else #elif #endif sono come le parole chiavi che usiamo normalmente IF ELSE ELSEIF , dove if vuol dire se

IF o #IF a=b SE a=b
facciamo questo : printf(" ciao ");

ELSE IF o #ELIF ALTRIMENTI +SE a>b
facciamo questo : printf(" arrivederci "); e seguitiamo nelle scelte

ELSE o #ELSE OPPURE
faccio questo perché a non è ne = ne > di b, invece abbiamo a < di b printf(" addio ");

l' ELSE IF o #ELIF possiamo inserirli più di uno infatti il IF finale dice segue

nelle scelte finche non trova ELSE ultima scelta per poi chiudere con la graffa } per l' IF normale mentre #ENDIF serve proprio per chiudere le scelte che fa gli if del preprocessore o precompilatore

#ERROR e #WARNING sono direttive che serve per segnalare errori o di avvertimento vanno usate sempre con le direttive #IF ecc...

#LINE ti manda alla linea dove dice il compilatore alla sinistra come spiega qui ;

http://www.cquestions.com/2011/01/line-directive-in-c.html

Altri comandi non ho trovato nulla su #pragma

02) Variabili e costanti sequenze di escape le graffe

Prima di parlare delle variabili parliamo dell' apertura del programma nel 1° capitolo abbiamo parlato delle direttive per il compilatore che inizia il nostro listato si inizia col caricare i file header .h o altri file di supporto dopo questi #include sono inserite le dichiarazione delle variabili , costanti, e si definisce i Define che sono anche essi costanti poi si deve inserire i prototipi delle varie procedure o funzioni che vedremo altrove dentro questa sezione

Fatto tutto questo ecco che per iniziare il nostro listato, lavoro o alcuni lo chiama token bisogna inserire l' inizio vero e proprio del nostro lavoro ;

#include<stdio.h>
ecc...
variabili costanti prototipi

ecc... ecco il vero inizio int main () { per creare la graffa aperta { premere alt sinistro + il numero 123, (i numeri da usare deve essere quella della tastiera dei numeri e non quelli in alto)per la graffa chiusa alt sinistro +125 } l' ultima graffa chiude il programma

ecco i commenti questi sono inseriti dentro 2 caratteri inizio /* */ fine, altro medo di inserire i commenti è // questo tipo di commento va posto solo su una riga e non c'è fine i commenti possono essere inseriti ovunque

ricordiamo la parte importante è il punto e virgola tanto se non viene inserito il compilatore da errore nella riga sottostante poche volte non vanno inseriti e questo lo vedremo man mano che camminiamo in avanti

int main() {

/* questa barra e asterisco

racchiude i commenti su più righe tutto quello
che sta qui dentro il compilatore non lo legge
 alla fine si inserisce
i 2 caratteri al contrario */

corpo del programma

system ("pause") ;

return 0;

}

segue con le procedure e funzione che abbiamo inserito su prima dell' inizio con i prototipi

Ecco la parte che chiude il programma , il system ("pause") serve per fermare la schermata dove ti dice " premi un tasto per continuare ... " e la graffa } serve proprio per chiudere il nostro listato il return in alcuni programmi in C nemmeno serve come non serve int vicino al main dove inizia il programma

Ricordate alla scuola trovate il lato a , il lato b se noi facciamo a x b l'area di un quadrato il Pc ragiona come noi gli serve le variabili ovvero i contenitori dove contenere questi numeri tanto per trovare l' area di un francobollo o di una piazza quadrata dobbiamo sempre fare a x b che sono i lati cosi troviamo l' area del quadrato

Ecco che il Pc il personal computer elabora i dati dentro queste variabili per lui 1 o 1 milione è uguale è solo un calcolo da fare, secondo i tipi di variabile , perché il Pc vuole sapere prima che risposta desideriamo avere un numero con la virgola o un numero intero ? ma questo ci deve interessare anche noi esempio se parliamo di età di una persona ci basta un numero che ci consuma poca memoria , inutile dire al Pc voglio un numero int long che qui fa calcoli di milioni per noi ci basta il più piccolo secondo questa tabella copiatela vi sarà molto utile oppure cercate in internet valori delle variabili in C vedi foto 1:

Le variabili e le costanti prima di usarle devono dichiarate sempre secondo lo schema della foto per dichiarare una variabile o costante prima cosa non può utilizzare le parole chiave del programma che sono parole riservate poi non possono iniziare con un cancelletto #

Se ci serve conoscere l' età di una persona fisica si richiama int , come vedete il float è da 4 bytes contro i 2 di int ;

possibilmente far capire cosa deve esprimere il dato esempio di prima dobbiamo

trovare l' area usiamo un numero adeguato che potrebbe essere benissimo un float per le virgole perché le misure può essere benissimo con le virgole esempio 3.45 centimetri se usiamo i numeri interi certamente non misuriamo più la virgola ecco un possibile idea dichiariamo 3 variabili dello stesso tipo :

long lato_a , lato_b , area ;

Se conosciamo i dati possiamo anche inizializzare le variabili o costanti, cosa vuol dire? dare un valore alla stessa variabile possiamo fare:

float lato_a = 3,5, lato_b = 2,50 , area;

Le variabili di diverso tipo vanno sempre scritte nelle righe nuove oppure scrivere le nuove dopo il (;)Possiamo anche scrivere una variabile per ogni riga

float lato_a; int anni ;

float lato_b = 12.3; int anni = 22 ;

float area ;

Costanti per le costanti e tutto uguale alle variabili solo che vanno dichiarate con la parola
"const " i valori delle variabili cambiano sempre , mentre il valore della costante non cambia mai un esempio è l' iva sulle fatture rimane sempre un numero fisso , per farle notare meglio scriverle in maiuscolo, poi tutto rimane come le variabili es :

const float IVA1 = 22 , IVA2 = 10 ;

ora noi per trovare l' area del quadrato facciamo lato a per lato b bene il Pc fa uguale solo che prima dobbiamo dargli un contenitore dove inserire questo valore infatti area :

area = lato_a * lato_b ;

* l' asterisco serve per fare le moltiplicazioni e la barra / quella nei tasti numerici è il diviso

per fare le graffe { si deve premere insieme alt + 123 dei tasti numerici o alt+125 per l' altra } oppure nella mia tastiera altGR + shift + parentesi quadra aperta o chiusa

Per usare le costanti o variabili bisogna imparare un' altro schema foto 2, nella foto

2 troviamo anche le sequenze di escape sono dei caratteri o meglio comandi che il compilatore altrimenti non capirebbe un esempio per stampare le virgolette " il compilatore non capisce se abbiamo chiuso un printf o se abbiamo sbagliato invece con " \\ " lui, il compilatore capisce che cosa deve fare, non è questo compilatore del Visual che è complicato anzi questo ci aiuta con quella finestra di (intellisense) che si apre quando digitiamo le parole tutte queste sequenze sono per tutti i compilatori, noi anziché imparare una lingua dobbiamo imparare un linguaggio del nostro computer il C è un linguaggio di basso livello perché è più vicino al linguaggio macchina e ha pochissime istruzioni o macro da eseguire , per macro si intende una sequenza di comandi i file header h che invochiamo possono essere come delle macro come comandi per il nostro programma già creati , oppure macro che possiamo creare noi tramite le funzioni che sono uguali hai fogli di calcolo anche li si invoca il linguaggio vb per creare macro

per ora qui chiudiamo e apriamo un' altra parte altrimenti facciamo casini comunque ricordate le 2 foto sono importanti per decifrare le varie variabili e costanti sia per creare , sia poi per leggerle pertanto copiatevi questi indirizzi per accedere alle foto ;

foto 1 per dichiarare le variabili e le costanti :

foto 2 per richiamare le variabili e costanti dai comandi printf e scanf , in più abbiamo le sequenze di escape :

03) Usiamo le variabili, le costanti e il comando #define

La prima cosa che dobbiamo ricordare sono come già detto i 2 schemi relativi alle variabili e costanti prima di dichiarare queste bisogna dirgli che tipo è , pertanto ecco dove sono reperibili le 2 foto :

foto 1 per dichiarare le variabili e le costanti per richiamare le variabili e costanti dai comandi printf e scanf, in più abbiamo le sequenze di escape in uso con il C tenetevi queste foto alla portata di mano servirà molto spesso :

http://im4.freeforumzone.it/up/49/65/1573108800.jpg

Foto 2 operatori aritmetici e di confronto in uso con il C :

http://im4.freeforumzone.it/up/49/66/1283067126.jpg

Ora possiamo iniziare :

Le variabili abbiamo detto sono dei contenitori dove vengono inseriti i dati, il Pc

non fa altro che andare in quella scatola e leggere cosa c'è scritto , queste scatole secondo quello che contiene possono essere di diverse dimensioni , noi dobbiamo dirgli con il programma che cerchiamo di creare cosa deve fare la variabile, cosi la variabile va dichiarata semplicemente, uso il rosso per le scritture del programma :

```
int somma, contenitore_a , contenitore_b ;
```

ora noi per fare la somma dobbiamo fare solo il +pertanto è facile, anche se poi abbiamo un altro schema da vedere per altre cose ma ora è presto , cerchiamo i valori inserisco anche commenti con il verde come fa il nostro compilatore si usa come detto // se è su una linea, oppure /* */ quando si commenta su più righe :

```
printf (" quanti pezzi ci sono nel contenitore a inserire un numero : \n ");  /* chiediamo di inserire un valore, il \n ci manda a capo nella nuova linea, cosi troveremo il cursore sotto la frase di richiesta, qui ho scritto un commento su più righe lo aperto con /* e ora lo chiudo con : */
```

ora con scanf chiediamo veramente il valore, vedete ci serve per richiamare int il formato decimale foto2 %d che serve per tutti i numeri decimali

```
scanf ("%d", contenitore_a);  //ora dobbiamo inserire il numero intero e richiediamo per il secondo contenitore
```

come vedete il secondo commento e solo sulla stessa riga è aperto con

\\ ora facciamo la prossima richiesta :

```
printf (" quanti pezzi ci sono nel contenitore b inserire un numero : ");
scanf ("%d", contenitore_b);
```

Qui ora nel printf se avete fatto caso non ho inserito il \n pertanto adesso il cursore rimane al fianco dei 2 puntini (:)
ora facciamo la somma :

```
somma = contenitore_a + contenitore_b ;
```

```
printf ("\ndentro i 2 contenitori c' erano tot pezzi %d ", somma );
```

Somma il numero viene stampato proprio nella parte dove c' è %d si poteva scrivere anche diversamente e il numero appare prima di pezzi li dove c'è %d o altre variabili di richiamo , il \n con la parola e tutto attaccato in modo che quando stampa il cursore rimane nella prima colonna :

```
printf ("\ndentro i 2 contenitori c' erano  %d  pezzi   ",  somma );
```

Semplice vero se si usava le costanti era uguale vanno trattate come se fosse normali variabili solo che prima di dire che tipo è va scritto che parliamo di costanti , per costante si intende un numero che per tutto il programma rimane sempre lo stesso, cosi quando dobbiamo cambiare quel numero ci basta cambiare quella costante o #define facciamo un esempio se quel numero è l' IVA quando si deve sostituire il tasso basta sostituire solo quel numero e tutto il programma funziona bene ecco come si dichiara una costante e la #define ;

```
#define  IVA  12 ;

const int  Iva = 22  ;
```

04) Ora vediamo i confronti tra le variabili

Come già detto il Pc l' unica cosa che capisce è lo 0 oppure 1, ovvero qui c'è la corrente oppure no , cosi lui è come noi se stiamo dentro casa c'è le tapparelle serrande aperte oppure sono chiuse proprio l' altro giorno il 30 aprile 16 si è festeggiato l' anniversario del centenario della nascita di Claude Shannon che è il padre del sistema binario questo sistema di conteggio apri le porte alle macchine che oggi noi usiamo lui uso dei semplici interruttori per sostenere la teoria di Boole che parlava proprio dei segnali di fumo che usava i miei antichi amici bene questi due grandi scienziati hanno aperto la porta all' informatica

La Cpu il vero cuore del computer è diviso in settori un settore c'è la memoria dove noi inseriamo le variabili ecc… poi c'è l' ALU che è la parte che ragiona infatti qui fa proprio le operazioni aritmetiche logiche di controllo quella parte che oggi vediamo

Cosi noi abbiamo dentro le scatole o Memoria dei numeri questi numeri ora grazie l' ALU possono essere confrontati oppure addizionati divisi ecc… questa è la parte più importante della nostra macchina o Pc il pc manda dei segnali sempre con i fumi, se cè il fumo è 1 altrimenti è 0 nell' insieme di 8 bit, otto segnali consecutivi corrisponde esattamente al numero 255 cosi il Pc con solo otto lampade tutte accese o diversamente accese può accedere fino a 255 diverse posizioni otto bit si chiama Byte

Se queste lampade accese spente sono inserite dentro l' ALU può lavorare diversamente secondo schemi diversi in diversi ambiti ma io mi fermo qui noi parliamo di programmazione dove i comandi che il Pc conosce sono secondo le foto che dovete copiare come memorandum ci sono maggior parte dei comandi usati in C :

foto 1 per dichiarare le variabili e le costanti per richiamare le variabili e costanti dai comandi printf e scanf, in più abbiamo le sequenze di escape in uso con il C tenetevi queste foto alla portata di mano Vi servirà molto spesso special modo ora all' inizio :

http://im4.freeforumzone.it/up/49/65/1573108800.jpg
Foto 2 operatori aritmetici e di confronto in uso con il C :
http://im4.freeforumzone.it/up/49/66/1283067126.jpg

Per noi ora interessa la seconda Foto dove troviamo tutti i comandi di confronto tra le due variabili cosi abbiamo :

< MINORE , > MAGGIORE, <= MINORE UGUALE, =>UGUALE MAGGIORE , !=DIVERSO,== UGUALE QUESTO DA NON CONFONDERE CON = USATO PER ASSEGNARE UN VALORE A= 8 OPPURE B = A CHE PRENDE IL VALORE DI A ANCHE B DIVENTA OTTO , MENTRE A==B CONFRONTA SE I 2 SONO UGUALI A UGUALE B

Abbiamo l' operatori aritmetici + - * / % quest'ultimo ci restituisce il resto su una divisione tra 2 interi per il Pc 5/ 2 ci da 2 tra interi e usando 5%2 ci restituisce 1 ecco un programmino funzionante ora guardate anche le posizioni delle varie variabili la variabile A è globale ovvero si legge per tutto il programma , mentre la variabile B è letta solo dentro il blocco iniziale tra le 2 graffe inizio e fine , la variabile C è letta solo dentro il blocco delle 2 graffe di IF :

```
 // Prove varie    // commenti su una riga
 /*commenti su più righe  ( si apre e si chiude con (/* */) il C normale conosce solo
questo commento il nostro compilatore VS riconosce entrambi */

#include<stdio.h>
#include<stdlib.h>

int a = 5;   // VARIABILE GLOBALE PER TUTTO IL PROGRAMMA

main()  {

int b = 2;  /* VARIABILE LOCALE SOLO PER DENTRO LE GRAFFE DI
               APPARTENENZA  */
   if (5 > 2)
       {
       int c; /* LOCALE SOLO DENTRO LE 2 GRAFFE dentro questo IF */
       c = a / b; // divisione tra le variabili a e b il riporto viene messo nella
                  //variabile c
       printf("C e' = %d , con il resto %d\n ", c, a%b); // a%b ci restituisce  1

}
        /* se qui cerchiamo la variabile c non esiste, perché è dentro l'altro blocco
*/
      printf("\nprova a++ e ++a con 2 stesse operazioni \n ");
```

```
        double aa = 3, bb ;

         bb = 20 + 1 / aa++;

        printf(" a++ bb= %f\n", bb);

        bb = 20 + 1 / ++aa;

        printf(" ++a b= %f\n", bb);
system("pause");    }
```

Nella foto degli operatori trovate A++ o A-- possiamo avere anche --A o ++A ma i risultatati possono essere differenti A++ fa la somma A+1 invece ++A fa 1+A e con una operazione può portare scompiglio un esempio come scritto nel programma su A= 3, con (20 + 1 / a++ = 20, 33) contro (20+1 /++a = 20,20) qui il ++ viene calcolato diversamente divisione e porta risultati errati

05) TIPO CASTING E CONFRONTI OERATORI LOGICI AND, NOT , OR

L' ultima volta abbiamo parlato del comando %modulo che ci restituisce il resto su una divisione tra numeri interi quell' errore di dividere 5/2 = 2 può essere evitato cambiando il tipo di assegnamento della variabile , già per se il compilatore lo fa da solo quando trova numeri di tipi diversi questo lavoro si chiama il casting se il compilatore trova :

1° operando	2° operando	Risultato restituito
long double	double	long double
double	float	double
float long	int	float
double long	int	double
int	char short	int
long int	short int	long int

Anche se il compilatore sa cosa deve fare possiamo sempre perdere l' informazioni meglio prevedere sempre prima come abbiamo detto se un intero INT 5/2 porta 2 anche se avessimo un Intero e un float o double il compilatore ci può restituire 2,00

Per fare il casting si usa cosi :se abbiamo A e B numeri interi mentre C è un tipo double o float cosi ci restituisce il numero preciso 5/2= 2,5 ecco come si usa il casting :

c= (double) a / (double) b

Nella lezione scorsa si parlava dei confronti ora affronteremo altri comandi logici che sono AND && , NOT! , OR||, XOR^ , questi confronta il Vero =1 con il Falso =0

AND&&, se trova A e B entrambi Veri = 1 allora è Vero = 1 , se A è 1 e B è 0 allora è Falso = 0, se A è 0 e B è 1 allora è Falso = 0, se A e B sono 0 allora tutto è Falso = 0

OR||, se trova A e B = 1 Vero allora è Vero = 1, se trova A = 1 e B= 0 oppure trova A = 0 e B = 1 allora tutto rimane Vero = 1 , se trova A e B = 0 restituisce 0 =Falso

XOR ^ , ovvero se trova A e B = 1 Vero allora è Falso = 0 , se A=1 e B= 0 oppure A= 0 e B= 1 restituisce Falso = 0 , se A e B sono 0 allora rimane 0 = Falso

NOT! Se a = 1 restituisce Falso = 0 , se A = 0 restituisce Vero = 1

Abbiamo già visto l' operatori razionali e logici questi fanno i confronti se A e B sono uguali ==, maggiori >, o diversi !=, oppure minori uguale <=, l' operatori logici è AND,OR,XOR e NOT che abbiamo appena visto poc'anzi, ci sono altri uguali per assegnare se li vede molto spesso sono usati nel C++ ma il nostro compilatore li conosce vediamoli è operatori assegnazione = A= 3, e possono essere diversi:
(+=, -=, *=, /=, %=, >>=, <<=, &=, ^= ,|=) possiamo usarli come :

A += 3 equivale dire A= A +3 , A*=B equivale dire A= A * B e cosi via quelli scritti in rosso sono operatori diversi che lavora sui bit e sono **(<<SHIFT SINISTRO , >>SHIFT DESTRO, & AND, |OR, ^XOR, ~NOT**) (~tasti ALT+126) questi operatori Bit Bit lavorano solo con i numeri interi poi non ho trovato come usarli forse lavorano uguale ma fa i controlli su i bit anziché sulle variabili da un testo ho trovato solo questo che copio e incollo viene dal testo :

Guida pratica alla Programmazione
Autore: BlackLight < blacklight@autistici.org >
rilasciato sotto licenza GNU GPL 3, copyleft 2005-2008
Un'altra operazione logica messa a disposizione dal C è lo SHIFT.
Immaginiamo di avere una variabile int i = 4; scritta in binario (facciamo per comodità a 4 bit) sappiamo che equivale a 0100. Fare uno shift a sinistra di 1 bit (la scrittura in questo caso è <<) equivale a spostare tutti i bit di un posto a sinistra: la
nostra variabile binaria da 0100 diventa quindi 1000, quindi i da 4 diventa per magia
8! Una cosa degenere in C si scrive così:
int i = 4;
i = i << 1; // Faccio lo shift a sinistra di 1 bit
C'è anche lo shift a destra, il simbolo è >>. Ad esempio, se facciamo uno shift a destra di 1 bit di i, questa variabile da 0100 diventa 0010, quindi da 4 diventa 2:
int i = 4;

i = i >> 1; // Faccio lo shift a destra di 1 bit

06) Dopo i confronti con operatori non può mancare IF ELSE IF

Intanto vediamo la formulazione come si usa :

if (condizione) istruzione;

else if (condizione) istruzione;

else if (condizione) istruzione

.

.

else istruzione

Come già visto con #IF comando del preprocessore, ora vediamo il vero confronto fatto da IF questo comando può essere sostituito dal ? come detto già IF vuol dire SE e allora SE A > B facciamo questo, poi c'era #ELSE, anche ora c'è ELSE che diceva OPPURE facciamo quest'altro , anche qui esiste il ELSE IF ecco un esempio pratico dell' l' uso di IF completo con le varie scelte … un If semplice con una scelta la base :

---------- Programma ----------

```
#include<stdio.h>
#include<stdlib.h>

main()
{
	int a = 3, b = 5, c = 8, d = 10, sca; //sca è una variabile = scatola

/* ho delle scatole a,b,c,d  Scatole = Variabili  con dentro dei prodotti ora vediamo il suo contenuto usando IF semplice   */

	printf("Abbiamo 4 scatole con diversi prodotti\n quale scatola desideri vedere \n(a=1, b=2 ,c=3 ,d=4 ) scelta ? ");

	    scanf("%d",&sca);

	if (sca == 1)

	        printf(" Qui abbiamo la variabile = scatola A,\n al suo interno c'e':%d pezzi \n", a);



	printf(" \n ByMpt-Zorobabele \nFine istruzione IF \n");
```

```
        system("pause");
}
```

---------- Fine programma

Ora lo faremo con 2 scelte A e B introducendo solo un ELSE e cambio solo il corpo dell' IF ecco l' esempio inserisco solo l' IF e ELSE semplice :

```
 if (sca == 1)
                printf(" Qui abbiamo la variabile = scatola A,\n al suo interno c'e':%d
pezzi \n", a);

          else

                printf(" Qui abbiamo la variabile = scatola B,\n al suo interno c'e':%d
pezzi \n", b);
```

Ora vediamo un IF annidato con 2 IF e un ELSE su ogni IF va la condizione ora possiamo fare 3 scelte scatola A,B,C (MI RACCOMANTO DENTRO LA SCELTA SI DEVE USARE IL CONFRONTO == E NON = CHE SI USA PER L' ASSEGNAZIONE) anche qui inserisco solo il corpo dell' IF , IF, ELSE e inserisco solo questa funzione tra il vecchio IF e ELSE :

```
 if (sca == 3 ) + condizione
```

ecco il programma modificato :----------

```
if (sca == 1)
                printf(" Qui abbiamo la variabile = scatola A,\n al suo interno c'e':%d
pezzi \n", a);

        if (sca == 3 )
                printf(" Qui abbiamo la variabile = scatola C,\n al suo interno c'e':%d
pezzi \n", c);

        else

                printf(" Qui abbiamo la variabile = scatola B,\n al suo interno c'e':%d
pezzi \n", b);
```

Quando chiamiamo IF – ELSE ecc… il compilatore stesso ci inserisce le graffe noi scriviamo dentro queste graffe e rispettate le posizioni ora vediamo tutto IF ELSE IF e ELSE per tutte le 4 scelte inoltre uso il DO WHILE (scelta) per far girare il piccolo programma , 2 cose importanti nella programmazione in C è == e non = come già detto poi ricordarsi nello SCANF di inserire & scanf("%d",&s) importante che la "%d" non ci deve essere spazi " %d "cosi è sbagliato , altrimenti quando mettiamo

la nostra scelta il programma aspetta il secondo numero o scelta e il compilatore non dice nulla nella compilazione, ecco il testo integrale del programmino :

```
#include<stdio.h>
#include<stdlib.h>  // libreria che legge il system(pause)

int s; //variabile che userò nella scelta  per il DO WHILE

main()
{
	int a = 3, b = 5, c = 8, d = 10, sca; //sca è una variabile = scatola
	/* ho delle scatole a,b,c,d  Scatole = Variabili  con dentro dei prodotti ora vediamo il suo contenuto usando IF semplice   */
	do
	{

	printf("Abbiamo 4 scatole con diversi prodotti\n quale scatola desideri vedere \n(a=1,-
		-b=2 ,c=3 ,d=4 ) scelta ? ");
	scanf("%d", &sca);

	if (sca == 1)
	{
		printf("Scatola A dentro c'e' il numero %d ", a);
	}
	else if (sca == 2)
	{
		printf("Scatola B dentro c'e' il numero %d ", b);
	}
	else if (sca == 3)
	{
		printf("Scatola C dentro c'e' il numero %d ", c);
	}
	else
		printf("Scatola D dentro c'e' il numero %d ", d);


    printf("\n Desideri vedere altra scatola ? S=1 o N=0 ");
		scanf("%d",&s);

} while ( s == 1 );

	printf("\n\n [ ByMpt-Zorobabele ] \n\nFine istruzione IF \n");
	system("pause");}
```

FINE PROGRAMMA d'esempio per l' utilizzo di IF ELSE IF

07)SWITCH CASE

SWITCH CASE è molto più semplice di IF il suo utilizzo è :

```
switch (variabile)
{
case costante 1: sequenza di istruzioni break;
case costante 2: sequenza di istruzioni break;
.
.
case costante n: sequenza di istruzioni break;
default       sequenza di istruzioni
}
```

Dentro la variabile di SWITCH ci va solo la variabile senza il confronto va fatto dentro le costanti che fa riferimento alle varie opzione riprendiamo il programma di IF ELSE IF alla discussione 06) e vediamo la differenza e la semplicità dello SWITCH CASE BREAK questo break fa uscire dall' istruzione CASE ecco il programma integrale di SWITCH stesso programma integrale di IF :

---------- inizio programma ----------

```
#include<stdio.h>
#include<stdlib.h>  // libreria che legge il system(pause)

int s; //variabile che userò nella scelta  per il DO WHILE

main() {

        int a = 3, b = 5, c = 8, d = 10, sca; //sca è una variabile = scatola
        /* ho delle scatole a,b,c,d  Scatole = Variabili  con dentro dei prodotti ora
                vediamo il suo contenuto usando IF semplice   */
        do
        {

        printf("Abbiamo 4 scatole con diversi prodotti\n quale scatola desideri vedere
                  \n(a=1, b=2 ,c=3 ,d=4 ) scelta ? ");
           scanf("%d", &sca);

         switch (sca )// questa variabile va confrontata con i case 1: case 2: ecc..
        {
        case 1:
```

```
          printf("Scatola A dentro c'e' il numero %d ", a);
                    break; // inserire sempre il break questo comando fa uscire dal
case
        case 2:
          printf("Scatola B dentro c'e' il numero %d ", b);
           break;

        case 3:
         printf("Scatola C dentro c'e' il numero %d ", c);
              break;

        default:
                    printf("Scatola D dentro c'e' il numero %d ", d);

        }

    printf("\n Desideri vedere altra scatola ? S=1 o N=0 ");
              scanf("%d",&s);

} while ( s == 1 );

        printf("\n\n [ ByMpt-Zorobabele ] \n\nFine istruzione IF \n");
        system("pause");
}
----------------------    fine programma   -------------------------
```

Se si ha molte opzioni meglio usare questo che IF SWITCH è molto più veloce da strutturare

08) Istruzione per i cicli DO WHILE , WHILE , FOR

L' istruzione DO WHILE l' abbiamo vista nel programma precedente questo comando fa si che un programma gira finche non trova la condizione falsa , questo è l' unico che confronta la condizione per ultimo, pertanto fa sempre un primo giro prima di conoscere la condizione ecco il suo formato semplicissimo :

```
do    // inizio del DO WHILE
{istruzione    //qui dentro le due graffe  tutte le istruzioni che devono seguire
}while (condizione)  /* la graffa e WHILE chiude l' istruzioni la condizione elabora
se ritornare al  DO */ oppure uscire  secondo la scelta che si è fatto in precedenza
esempio a == 0  allora esci altrimenti rimani sempre dentro il ciclo
```

Altro ciclo è l' inverso il WHILE ma senza il DO ecco la sua sintassi più semplice di tutti :

while (condizione) istruzione

ecco un esempio il nostro compilatore ci da :

```
while (true)    //dove qui il true vero ci deve essere la condizione esempio (contatore
 { }                    //   <10) dentro le parentesi graffe c'è tutto il ciclo
Int contatore = 0; //mettiamo nel programma un contatore che conta i passaggi
While  (contatore < 10)
{
    istruzioni …
   contatore++ ; // il contatore va sommato per contare i cicli fatti
}
```

Ora vediamo il ciclo del FOR è un po' più complesso ma semplice già il compilatore ci mostra la sintassi che ecco come lo scrive ;

```
for (size_t i = 0; i < length; i++)
        {
Istruzioni
        }
```

Nella parte iniziale (size_t i = 0 ;) questa è la inizializzazione della variabile contatore pertanto i= 0 togliere size_t che vuol dire dimensione o grandezza ,possiamo mettere questa dimensione anche = 5 o 10 secondo quello che ci serve , la seconda parte è (i< length ;) anche qui la parola length che vuol dire lunghezza va via e rimane la vera istruzione ovvero il confronto della variabile I con un il numero dei cicli che desideriamo far fare con i <= 10 cosi il contatore quando arriva al 10 si ferma ora la parte finale è il contatore (i++) questo contatore conta la variabile i +1 come abbiamo già visto ma possiamo anche sostituirlo con un conteggio diverso i+5 possiamo inserire anche delle doppie istruzioni nella stessa sequenza del FORe ecco un esempio funzionante :

```
int x,y;   // dichiarazione delle 2 variabili x e y che useremo dentro il for
for(x=0, y=0; x+y<=50; x++, y+=3)
 printf("- %d - ",x + y);
```

la prima parte si inizializza i contatori x e y = 0, poi x e y vengono sommati tra loro questa somma deve arrivare <= 50 , ora x viene sommato solo di 1 mentre

y è sommato di 3 anche dentro il printf viene sommati x e y, ecco cosa ci restituisce il tutto a video;

```
0 - - 4 - - 8 - - 12 - - 16 - - 20 - - 24 - - 28 -

------------------------------

 [ ByMpt-Zorobabele ] Fine istruzione

Premere un tasto per continuare . . .
```

09) Istruzioni di salto return, break, continue , exit

Una istruzione che non si usa più è il goto era semplicissima :

goto etichetta ;

dove etichetta poteva essere in qualsiasi parte del programma e con il goto si saltava su quel punto , ma poi troppe etichette e goto faceva del programma un casino da non capirci più nulla allora l' ANSI l' agenzia di standardizzazione ha abolito il goto per tutti i linguaggi di programmazione

il Return fa uscire con un risultato da una funzione anche il Main () è una funzione, è la funzione principale del programma normalmente si usa return 0 , poi la vedremo meglio in uso dentro le funzioni

Il comando Break già lo abbiamo usato dentro con il comando lo SWITCH CASE BREAK ci faceva uscire dalle varie condizioni questa chiave non è fondamentale se non c'è le istruzioni vengono eseguite tutte sequenzialmente dentro lo stesso SWICTH

Break è possibile usarlo dentro i comandi che fa girare i programmi con i loop giri continui tipo istruzioni FOR, WHILE e DO WHILE quando il programma gira e trova il Break esce da quella situazione e si porta alle prossime istruzioni che segue dopo le graffe

Se ci sono più loop annidati il break fa uscire solo da quel FOR annidato e riprende dal prossimo FOR che trova

Il comando o istruzione CONTINUE è uguale al break solo che fa continuare i giri dentro i loop i cicli continui, questo comando non può essere usato dentro lo SWICTH

Dentro i cicli di WHILE e DO-WHILE se si incontra il CONTINUE fa si che l' istruzione riparte dal controllo che viene rivalutato se vero allora fa il giro altrimenti esce dal ciclo , se ci sono cicli annidati il CONTINUE riparte dallo stesso ciclo interno

L' istruzione EXIT fa uscire dall' intero programma

Tutte le vere istruzione che abbiamo visto sono :

per i salti : RETURN, GOTO, BREAK, CONTINUE, EXIT

per i loop : FOR, WHILE, DO-WILE

per controllare : IF , IF ELSE, IF ESE IF ELSE, SWITCH

Qui ho detto che il MAIN è una funzione , la funzione principale del programma, prossimamente vedremo le Funzioni

10) Le FUNZIONI

Dentro le funzioni possiamo metterci il lavoro sporco , per poi richiamarle dentro il programma o funzione principale il MAIN con un semplice (nome ();) come potete capire anzi che fare delle grandi operazioni dentro il main è bene farle dentro le funzioni poi le funzioni evita di fare i doppi lavori se serve un' istruzione più di una volta allora la funzione basta richiamarla dove desideriamo, anche il printf e scanf sono funzioni del compilatore che noi richiamiamo ogni file .H eader contiene le sue funzioni che poi noi richiamiamo dentro i nostri programmi proprio per questo che dentro questo linguaggio C c'è pochissimi comandi o istruzioni grazie alle funzioni già costruite

La prima cosa che si fa per le funzioni , devono essere dichiarate con il prototipo in globale come si dichiara le variabili o le costanti, per dichiarare una Funzione abbiamo:

```
tipo_restituito    nome_funzione     (parametro1, parametro2);
```

Questo prototipo serve per dire al programma che c'è questa funzione che usa i parametri 1 e 2 e mi viene restituito un tipo di calcolo i tipi sono proprio il tipo di variabili int, double, float o void che è vuoto , senza questa dichiarazione sarebbe come utilizzare una variabile senza averla prima dichiarata cosi il compilatore da errori se non viene inserito nessun tipo allora viene considerata int per uscire da queste funzioni non void si usa il RETURN come già detto le variabili o le costanti globali vanno inserite prima del MAIN () e dopo i comandi del preprocessore ecco come si dichiara una funzione vediamola da vicino il primo int è quello che ci restituisce (tipo_restituito) la funzione somma , (somma è il nome della funzione) (int , int) sono i (parametri 1 e 2) che la funzione prende per fare i suoi calcoli

```
int somma (int, int );
```

Creata questo prototipo della funzione il compilatore stesso VS2015 ci fa creare la funzione e la colloca alla fine del programma generale giù in fondo dopo la graffa

che chiude tutto il programma, (questo se il (main () { istruzioni }) è stato già creato), altrimenti viene collegata direttamente sotto la dichiarazione della stessa e il compilatore da errore perché non vede il programma principale , costruita la funzione prototipo vedrete il nome della funzione sottolineata in verde , portate il mouse sopra il nome compare una lampadina con la scelta (crea la definizione) questo fatelo dopo che avete inserito il programma principale cosi create la vera funzione che è il nome del prototipo con 2 graffe sotto , nella parte parametro 1e 2 dobbiamo inserire il nome delle variabili se non lo abbiamo fatto su nel prototipo e questo è il passaggio per valore , possiamo anche fare il passaggio per riferimento che vedremo poi con i puntatori

Al compilatore gli interessa sapere su cosa lavorare, ovvero deve sapere che tipo di variabili o costanti usiamo int, double, float, oppure void vuoto , se non scriviamo nulla il compilatore considera tutto int e lavora su i numeri interi ecco un esempio di una funzione vuota inserisce solo una intestazione quando chiamiamo la stessa funzione, nello stesso programma inserisco una seconda funzione per una somma di numeri reali float e qui come spiegato prima si esce dalla funzione grazie al RETURN ecco il programmmino funzionante :

<--------------------- inizio programma ---------------->

```
#include <stdio.h>
#include <stdlib.h>
                        // i 2 prototipi delle 2 funzioni inseriti nella
void intestazione();   // sezione globale che rimane tra #include e il main
float somma(float, float); // le 2 funzioni una è void l'altra tutto float

 main() {

        float aa, bb, cc;
        intestazione(); // richiamo della prima funzione che è l'intestazione

        //si richiede 2 numeri che dobbiamo sommarli dentro la funzione

        printf("\n Inseriamo un numero reale AA "); scanf("%f",&aa);

        printf("\n Inserisci un numero reale BB "); scanf("%f",&bb);

        cc = somma(aa,bb);  /* cc è il risultato che devo riprendere dalla funzione
per poi ristamparlo  qui sotto con la printf */

        printf("\n La somma e' di %.2f\n ", cc);


        printf("\n Fine ");

       intestazione(); //di nuovo la mia intestazione come su
```

```
	system("pause");	} //fine del programma main ora qui sotto troviamo le funzioni


void intestazione() // funzione dell' intestazione
{
	printf("  [ ByMpt-Zorobabele ] PROVA LE FUNZIONI LA SOMMA  AA + BB \n\n");

}

float somma(float a, float b) //funzione della somma guardate le variabili

{		// sono tutte diverse da quelle usate nella funzione

	float c;	// principale del main su si usa AA, BB,CC contro A,B,C

	c= a + b;

	return c ;  // come spiegato prima nella lezione 9 ecco il RETURN
}
```

<---------- fine programma ---------->

ecco il video stampato dove troviamo le 2 intestazioni , inserimento dei numeri12.30 e 5.70 e la somma 18 :

```
 [ ByMpt-Zorobabele ] PROVA LE FUNZIONI LA SOMMA  AA + BB
Inseriamo un numero reale AA 12.30
Inserisci un numero reale BB 5.70
La somma e' di 18.00
Fine   [ ByMpt-Zorobabele ] PROVA LE FUNZIONI LA SOMMA  AA + BB
Premere un tasto per continuare . . .
```

11) Array o Vettore e le Matrici le nuove variabili

Un vettore monodimensionale è un nuovo tipo di variabile costituito da tante scatole che viene definite sempre dello stesso tipo, il vettore o array monodimensionale il suo indice parte sempre da 0 zero per quante scatole abbiamo sotto lo stesso nome , intanto vediamo come si inizializza questa nuova variabile la chiamiamo vettore e è tipo int con 6 scatole interne dallo 0 al 5 :

```
int  vettore  [5] ;
```

Questa nuova variabile si inizializza cosi int il tipo che può essere o altro tipo poi abbiamo il nome vettore , dentro le [] va il numero di quante scatole è composto questo vettore o array monodimensionale , come detto il vettore parte sempre da 0 questo è l' indice del vettore quando noi facciamo riferimento al vettore senza indicazioni il programma va e vede solo il suo indice 0 , vediamolo da vicino lo stesso vettore da 5 scatole, ovvero i 6 contenitori che abbiamo inizializzato ;

vettore [0] = 12 ; // primo vettore l' indice con il suo contenuto numero int 12

vettore [1] = 100 ; // secondo vettore con il suo contenuto numero int 100

vettore [2] = 5 ; // terzo vettore con il suo contenuto numero int 5

vettore [3] = 20 ; // quarto vettore con il suo contenuto numero int 20

vettore [4] = 2 ; // quinto vettore con il suo contenuto numero int 2

vettore [5] = 40 ; // sesto vettore con il suo contenuto numero int 40

Ora se noi vogliamo leggere dentro il vettore 3 che è il quarto array dobbiamo scrivere vettore [3] per leggere il suo contenuto che è 20 ,viene trattato come una semplice variabile con il suo indice 0,1,2,3 ecc.. ecco come leggere il [3] vettore

printf(" dentro il vettore 3 c'e' questo contenuto %d ", vettore[3]");

Nel modo come ho fatto su con : (vettore [2] = 5;) ho anche inserito i dati dentro tutto il vettore, il vettore può essere dichiarato diversamente e nello stesso tempo inserire i dati esplicitamente :

int a[] ={1,3,5,11,112,6};

in questo modo abbiamo detto al Pc abbiamo un array di nome a formato da 6 elementi dove elemento 0 è = 1, ecc.. , elemento 5 è uguale 6

Possibilmente non utilizziamo molta memoria del Pc dichiarando array da 100 scatole quando ci basta 10 contenitori , infatti se noi diciamo : A[100]; il pc assegna la memoria per 100 contenitori , se consideriamo che un intero è 2 o 8 byte per un double, ogni byte è 8 bit per i 90 contenitori inutilizzati abbiamo sprecato minimo 1440 bit

Per accedere alle varie scatole o contenitori possiamo avvalerci del loop FOR ecco un programmino per caricare e leggere il vettore da 4 contenitori utilizzo le 2 funzioni per lo scopo carica() e leggi() :

```
#include <stdio.h>
#include <stdlib.h>
void intestazione();
int ay[3]; // 4 contenitori da 0-3 chiamati ay
void carica();  void leggi(); // 2 funzioni una carica l' altra leggi l' array
        // il tutto rimane nella parte globale in modo che le funzioni possa leggere

int main() {
```

```
	intestazione();		carica();
	intestazione();		leggi();
	intestazione();		system("pause");
}

void intestazione()
{
	printf(" -------------------------------------------------");
	printf("\n|  [ByMpt-Zorobabele  prova Array monodimensionale |\n");
	printf(" -------------------------------------------------  \n");
}

void carica()
{
	int i; // contatore i questa è una variabile locale viene usata solo qui
	for (i = 0; i <= 3; i++)  //giriamo per 4 volte  da 0 al 3
	{
		printf(" Carichiamo i nostri 4 contenitori elemento %d di ay: ",i);
		scanf("%d", &ay[i]);
	}
}

void leggi()
{
	int i;
	for (i = 0; i <= 3; i++)
	{
		printf(" Leggiamo i dati inseriti dentro l'array ay num %d =
%d\n",i,ay[i]);

	}
}
```

Ora parliamo delle matrici sono minimo 2 vettori con lo stesso nome infatti queste si chiamano array pluridimensionali perché ha più di una dimensione questi si dichiara ugualmente come i vettori normali:

int Array [3],[2] ; come sempre l' indice parte da 0 e conta tante colonne quanti sono descritti nella seconda dichiarazione ecco un esempio questo array bidimensionale va letto riga per colonna dove la riga è il primo numero le colonne sono il secondo numero , per la lettura di questa matrice si usa 2 indici , possiamo anche inserire i dati come spiegato array[2],[4]={

i numeri rossi fa parte del[2] {3,4,5,7}, tutti l' altri è del [4]

{12,25,70}

};

[0] [1] se noi dobbiamo andare a leggere l' ultimo dove risiede
[0] [102] [105] il numero 214 dobbiamo dare le sue coordinate come se
[1] [128] [13] giochiamo a battaglia navale array [2],[1] , per il numero
[2] [102] [214] 128 è riga 1 colonna 0 array [1],[0], arriva sempre riga per colonna , altro modo di inserire la matrice ;

```
int mat[2][4] = { 1, 2, 10,20,30 ,100,200,300 };
```

ricordiamo che per dichiarare le matrici i numeri dentro le quadre devono esserci sempre se non i numeri le variabili che fa riferimento ad un numero, non deve esserci mai 0 zero , i numeri 1,2 dentro la matrice fa riferimento per la parte [2] tutti l' altri è per la [4]

Le matrici dicono si usa poco allora nessuno spiega come usarle, comunque si deve usare per leggere e scrivere su di loro 2 indici contemporaneamente con 2 for annidati uno per la prima parte [2] e l' altro per la seconda lettura [4] ecc… se ci sono altri tridimensionale ecc… la guida del VS2015 sotto matrice un po' spiega per richiamarla F1 poi cerca

12) Array , le Stringhe i Puntatori

Il C non ha delle vere e proprio set per scrivere le parole , essi si rappresentano con le stringhe ovvero con l' Array monodimensionali il suo costrutto è semplicemente :

```
char nome [8]= "scatola" ;
```

Come nell' Array monodimensionale possiamo non inserire il numero di caratteri da usare ma è da considerare sempre un carattere in più di quello che abbiamo, questo carattere lo mette in automatico il compilatore e è un carattere nullo \0 che dice al compilatore dove finisce la stringa , come sempre l' array parte sempre da 0 fino all' ultimo carattere dove trova la chiusura della stringa \0

Altro modo di inserire o inizializzare una stringa di caratteri può essere come si fa con l' array dei numeri ovvero :

```
char nome [] = {'s','c','a','t','o','l','a','\0'} ;
```

forse è più comodo usare come su , anche qui se inizializzato l' array di stringhe non serve inserire dentro []il numero come per le matrici ,anche qui come per l' array normale si inizia a contare da 0, 1, 2, ecc

La stringa viene richiamata dal carattere %s , pertanto sulla funzione scanf e printf si inserisce la dicitura %s in scanf non serve più inserire il simbolo & che è il richiamo del puntatore pertanto scriveremo :

scanf(“%s”, nome_stringa);

Nel Compilatore C abbiamo una libreria chiamata <string.h> che dobbiamo includere , questa libreria ci offre delle funzioni per lavorare su questi array di stringhe vediamo quelle più usate :

STRCPY serve per copiare una stringa in un’altra

STRCAT concatena 2 stringhe

STRCMP confronta 2 stringhe

STRCHR cerca un carattere in una stringa

STRSTR cerca una stringa in un'altra stringa

STRLEN ci restituisce la lunghezza della stringa

Per altre funzione e non solo per <STRING> ma per tutti i file header (*.h) del C si può vedere in Wikibooks

https://it.wikibooks.org/wiki/C/Appendice/Librerie_standard

Il puntatore è l’ indirizzo fisico di una variabile puntata esempio abbiamo moltissime scatole che sono le nostre variabili queste scatole sono riposte nei scaffali i scaffali sono la memoria del Pc, noi con il puntatore troviamo subito la nostra variabile che usiamo , ovvero troviamo subito quella scatola riposto in quella fila di quel scaffale numerato sequenzialmente , come le variabili o l’ array prima dobbiamo dire che tipo di puntatore abbiamo che va in base alla variabile che dobbiamo leggere allora prima vediamo che tipo è int , double , float, char, trovato questo mettiamo il nome ecco il puntatore viene creato con l’ asterisco (*) allora abbiamo p= puntatore int , questo puntatore poi deve leggere una variabile int , inizializzo anche la variabile var = 5 che poi leggeremo ;

int p* ,var= 5;

ora il nostro puntatore p deve leggere la nostra variabile var che è inizializzata a 5 (var= 5) per fare ciò :

p = &var ;

Ora il puntatore p va dalla variabile var e legge in quale posizione è inserita la variabile var dentro il nostro magazzino ,se noi leggiamo il puntatore p e la variabile &var ci restituisce la posizione, se invece leggo *p e var mi restituisce il valore 5, la memoria che usiamo del Pc di solito è in formato sequenziale di posizioni e ogni variabile viene scritta in una parte libera della nostra memoria non è detto che per forza 2 nostre variabili siano inserite vicine sequenzialmente

L’ operatore & (e commerciale) è l’ operatore dell’ indirizzo legge l’ indirizzo o la posizione dentro la nostra memoria che utilizziamo, questo si applica solo ad contenuti che già esiste in memoria esempio alle variabili esistenti, una variabile non inizializzata (var = 5) può dare numeri a caso

L' operatore * (asterisco) questo è l'operatore che legge il valore reale della variabile che leggiamo ovvero *p ci produce 5 questo operatore si chiama de referenziare dall' inglese (dereferencing) , se noi facciamo (*p = 8 ;) , ecco che la variabile var in lettura cambia il suo valore e diventa 8 perché noi alla posizione della variabile var abbiamo cambiato il suo contenuto tramite il puntatore *p

Questi puntatori possiamo usarli anche con l' array [] sia che essi sia stringhe che sia numerici basta far riferimento i suoi indici per leggere l' indirizzo di :

main () { char sc[]="scatola"; int x; sc[7] = &x; printf (" L'indirizzo di sc[7] e' -> %d ",&x); system("pause"); }

Sembra che l' array e il puntatore siano diversi effettivamente nel C sono identici ma utilizzati diversamente il vettore o array quando vengono creati si crea un puntatore nella memoria che usiamo e si riserva tanto spazio per quanto è grande l' array secondo il tipo e la grandezza ecco 2 scritture uguali, abbiamo v vettore :

```
main (){    int v[]= {53,45,78};
                printf("leggiamo il secondo elemento del vettore : %d\n ",v[1]);
                printf("leggiamo il secondo elemento del vettore : %d\n ",*(v+1));
```

Passaggio dei puntatori verso le funzioni, abbiamo visto che è possibile lavorare con le funzioni il C ci permette di lavorare direttamente sulle funzione non elaborando i dati copia che sono volatili, ovvero che si cancellano appena chiusa la funzione , ma lavorando proprio con i puntatori lavorare proprio con l' indirizzo fisici reali delle vere variabili alla fine è meglio lavorare sulle funzioni con i stessi puntatori o array

Comunque non andiamo dentro oltre perché poi con il C++ non si usa più ,tanto poi vedremo altro diversamente

Prossimamente vedremo altri tipi di dati definiti da noi sempre qui in C

13) Definiamo noi i dati con typedef ancora array e matrici , enum, struct , union

Questi dati che noi possiamo definire sono ovvero dati legati tra di loro da una struttura ben definita la struttura viene collegata con typedef

Il typedef possiamo usarlo anche per le variabili semplici cosi rendendo il programma migliore in leggibilità esempio troviamo i lati di un rettangolo i lati li

chiamiamo lati a e b buono , ma con typedef possiamo fare diversamente vediamo la sintassi di typedef per capire come funziona ;

typedef tipo nuovotipo ; (← parte 1 // parte 2 →) nuovotipo nuovonome

dove tipo è il solito tipo della variabile int, float, char ecc…. nuovo tipo il nuovo nome che collega alla variabile e ora vediamo :

typedef int rettangolo_lato_A ; rettangolo_lato_A a;

ora la variabile rettangolo_lato_A viene rinominata in a che è il nuovonome ugualmente per il lato b ;

typedef int rettangolo_lato_B ; (<-parte 1) (parte 2 ->) rettangolo_lato_B b;

Tutto questo serve per avere una leggibilità migliore del programma la parte 1 e 2 della variabile dove viene rinominata con a o b può essere inserita sia sopra che sotto il main () { ,quello che cambia è la visibilità delle variabili se le variabili sono globali va posto sopra il main , altrimenti sotto il main qui le variabili non sono globali ma locali, si consiglia sempre prima del main cosi sono globali e possono essere lette anche dentro le funzioni, se non serve meglio sotto il main cosi gira solo dentro il main

Con il typedef possiamo creare direttamente i vettori di numeri e di caratteri logicamente, la sintassi è come su solo:

typedef tipo nuovo_tipo [dimensione] ; esempio typedev int vettore [5] ; vettore v; ora abbiamo v[0]v[1]v[2]..

Se ci serve più vettori della stessa dimensione e uguali tipo intero o altro possiamo fare nella 2 parte :

vettore v1,v2,v3 ; cosi abbiamo creato 3 vettori da 5 anzi che scrivere v1[5],v2[5],v3[5] ecc… inizializzare questi vettori è come al solito v1[]{5,123,46,400,1650}; se non viene inizializzato nelle quadre [n] va messo il numero o un riferimento ad una costante creata con un (#define n 5)

Con il typedef possiamo creare anche le matrici come su abbiamo;

typedef float matrice [3][4] ; matrice m;

Con il typedef possiamo definire nuovi dati che magari ha una lunghezza di tot caratteri una data di gg/mm/aaaa è composta di 11 caratteri con lo (\0) delle stringhe oppure i 17 caratteri di un codice fiscale , ugualmente per le targhe auto con 8 caratteri uscendo fuori dei vettori controllati

typedef char data [11] ; data dt;

Enum con enumerazioni possiamo associare dei numeri interi alle parole intanto ecco la sintassi :

enum nome_del_tipo {lista dei valori } variabile;

Ora vediamola in dettaglio :

enum settimana {lun,mar mer,gio,ven,sab,dom } giorni ;

il nome del tipo è settimana poi abbiamo i giorni dove ogni giorno è un numero intero, enum come i vettori conta da 0 pertanto : lun= 0 , mar=1 , mer=2 … dom= 6 se vogliamo cambiare i numeri basta fare ;

enum settimana {lun =1, mar, mer,gio, ecc..} giorni;

ora tutti quelli dietro il primo numero sono sequenziale partendo da 1 , se il numero era 50 per lun, mar era 51 ecc… se desideriamo i nostri numeri personalizzati basta inserirli manualmente ora possiamo vedere i confronti se mar è maggiore di lun oppure g1= lun ecc… in C non c'è il dato booleano ma possiamo costruirlo con l' enum semplicemente

enum bol {falso,vero}boolean ; anche in enum possiamo usare typedef cosi la parte 1 :

typedef enum {

lun= 1, mar,

mer,ecc:

} giorni ; (parte 2 -→) giorni gg;

Struct usiamolo con il typedef ci fa ritrovare sotto un nuovo tipo una nuova variabile strutturata e semplice da creare, noi come sempre abbiamo le scatole in riferimento una scatola del nostro negozio può essere un giocattolo oppure una lampada oppure un tavolo secondo la descrizione, se vendiamo le auto che ci serve la marca, il modello anno di immatricolazione, il costo ecc… se vendiamo i computer ci basta marca, modello, e costo noi possiamo strutturare tutti questi dati e questi dati sotto la struct struttura può uscire, il compilatore inserisce le 2 graffe in verticale io per comodità inserisco i dati in orizzontale, nella parte 1 fare attenzione su ogni nome c'è il tipo (char marca [20]) e i tipi possono essere diversi secondo le esigenze

typedef struct {

char marca [20]; char modello[20];int costo ;

} computer (parte 2 → (computer c;))

Diversamente senza typedef si poteva fare :

struct computer {

char marca[20]; char modello[20] ; int costo;

} ;

struct computer c ;

ora questo dato computer è collego ad una variabile nuova c , quando chiamo questa c e inserisco il punto ' . ' mi esce la lista che su avevo creato marca , modello, costo e li devo inserire i dati , queste nuove variabili sono collegate tra loro tramite il c del typedef, e alla struct del computer ora collegato nella seconda parte creiamo un vettore c di tanti posti computer c [10] ; cosi abbiamo una tabella di 10 vettori collegati mediante la struttura di struct immaginiamo una tabella dei nostri clienti cosa possiamo collocare come dati: il nome cognome, l' indirizzo, il telefono, la partita iva ,codice cliente, tipo pagamento, Iban , ecc…

Union è simile alla struct , vedo che nessuno la tratta nei testi di C che uso, ne parla solo la guida dello stesso compilatore, viene usato inserito dentro le istruzioni di struct potrebbe avere anche un overload un sovraccarico se lo si usasse con tipi differenti di variabili , perché union usa lo stesso settore della memoria per le varie variabili dichiarate in esso e non si può usare variabili di tipo int con un tipo double cosi si crea l' overload , la parte buona di union usando la stessa memoria usa meglio i settori della stessa memoria poi le troveremo anche nel C++ dentro la guida già ne parla di più per il suo nuovo uso dentro il C++11 è stato migliorato il suo uso ora noi rimaniamo in C perché come scritto in wikibook vedi dentro il forum la nuova cartella (Appunti sulla programmazione C++ (vuoto))e il suo unico collegamento guida dal C al C++, si consiglia la conoscenza del C per conoscere meglio il C++

14) Ancora puntatori, typedef, var. automatiche e statiche, malloc, free

Parliamo ancora del puntatore perché dopo i grandi dati che immagazziniamo grazie i record che sono le strutture che abbiamo visto nel capitolo precedente, ci può essere d' aiuto i puntatori e la funzione malloc non per ripetermi rivediamo i puntatori in breve il puntatore si costruisce come le variabili ma deve avere l' asterisco (*) prima del suo nome ecco la sua sintassi (p= puntatore) :

tipo_elemento *nome_puntatore ; int *p; abbiamo creato il puntatore p, ora dobbiamo dargli la vera variabile che deve leggere logicamente deve essere dello stesso tipo intera nel nostro caso per assegnare una variabile al nostro puntatore p basta inserire la e-commerciale (&) prima della variabile (v= variabile):

p= &v; abbiamo assegnato al puntatore p la variabile v se io scrivo *p = 12; , ecco che la variabile v prende il valore di 12 perché come si era detto in precedenza, (nel capitolo 12 dei puntatori) il puntatore legge l' indirizzo esatto di dove risiede la variabile dentro il disco nostro caso la variabile v , questo indirizzo lo legge grazie all' operatore dell' indirizzo & , stesso & che usiamo con la funzione scanf proprio per dargli un indirizzo di memoria e come detto il puntatore *p è lo specchio o alias di questo indirizzo , pertanto possiamo operare sulla variabile v

direttamente o indirettamente tramite il suo puntatore p un esempio inserisco il tutto su una riga per comodità testuale del forum se abbiamo :

```
int a = 8, v, *p ;  p = &v;  *p = 2;  *p = *p * a;  // il risultato del printf è : 16 e 16
printf("puntatore =[%d] variabile =[%d]\nByMpt-Zorobabele\n ", *p, v);
```

Nel capitolo 13 abbiamo parlato di typedef che serve per dichiarare variabili e tipi dati non primitivi , i tipi primitivi sono le semplici variabili …

Già l' array e tutto quello che possiamo costruire con typedef come le strutture struct e le unioni union questi sono tipi dati non primitivi, con typedef possiamo creare anche nuovi puntatori la soluzione è la stessa che avevamo visto che si divide in 2 parti parte 1 la vera dichiarazione , parte 2 il cambio del nome o dei nomi se dobbiamo creare più variabili uguali sintassi per i puntatori:

```
typedef tipo_elemento *nome_elemento;  nome_elemento nuovo_nome;
```

esempio :

```
typedef int *puntatore;  puntatore p;
```

Ora il nostro nuovo puntatore p è operabile rivediamo il programmino di prima ma ora usiamo il nuovo dato p del nuovo puntatore di typedef come vedete cambia solo la dichiarazione del puntatore :

```
typedef int *puntatore;  puntatore p;  int a = 8, v; p = &v; *p = 2; *p = *p * a;
printf("puntatore =[%d] variabile =[%d]\nByMpt-Zorobabele\n ", *p, v);
```

Fino ad ora abbiamo visto solo le variabili automatiche il suo tempo di vita di queste variabili è effettuato automaticamente dal sistema centrale se sono inserite prima del main sono globali sotto il main sono locali, se sono dentro le varie funzioni o blocchi {} sono solo locali per quelle funzioni blocchi quando si esce dalla funzione o blocco quella variabile viene cancellata

Ora parliamo delle variabili dinamiche , queste variabili siamo noi che le allochiamo con malloc e le de allochiamo con free da dentro la memoria centrale chiamata heap (heap =mucchio) , queste variabili non hanno un identificatore o nome ma possono essere attribuite per mezzo del loro indirizzo attraverso i puntatori , questi comandi malloc e free sono usabili tramite la libreria che dobbiamo inserire con #include<stdlib.h> la stessa libreria che riconosce il comando system , la sintassi di malloc (memory allocation) [p= puntatore] :

```
nome_puntatore = (tipo_puntatore *) malloc (sizeof (tipo_puntatore));
```

dove nome_puntatore è il nome p, tipo_puntatore è il tipo della variabile puntata int,double,float ecc… sizeof(t_p) calcola il numero di bit o byte che occupa il tipo puntatore se è int occupa 2 byte, un float o int long 4 , un double 8 byte vediamo un esempio di allocazione di memoria su variabili primitive vedi il file prova 2 allegato dentro il forum

http://im4.freeforumzone.it/up/49/24/944423184.txt

15) lavoriamo sui file, con una piccola visione su questo comando fopen

Qui faremo l' ultimo pezzo per il C per poi iniziare con il C++, per chiudere l' argomento del C non è possibile lasciare senza parlare di come salvare i file in C, la prima cosa in C i file sono due tipi i file binari e i file di testo per capire meglio sono i file tipo (*.txt) per gestire il FILE si deve fare 3 passaggi apertura , accesso, chiusura del file, il tutto va fatto tramite un puntatore al FILE

La prima cosa che dobbiamo sapere è come aprire il file *mode? in lettura , scrittura ,oppure continuare la scrittura dall' ultimo dato, secondo i casi dobbiamo inserire: per la lettura r = read, per la scrittura w = write, per scrivere continuando in fondo al file a = append se il file che dobbiamo aprire è binario con r,w,a va inserito b se non è scritto nulla il FILE è un file di testo, la parola chiave è fopen ecco la sintassi per aprire un file :

FILE *fopen (*name, *mode);

dove name è il nome del file prova.txt, dove mode è come aprirlo r,w,a +b se serve vediamo apertura in lettura : il puntatore FILE *fp; fp= fopen("prova.txt" , "r"); apertura in scrittura puntatore: FILE *Fp;

fp= fopen("prova.txt" , "w"); attenzione se si apre in scrittura il puntatore del file parte dalla posizione iniziale, cosi dati già scritti vengono persi perché si sovrascrive con i nuovi dati, mentre con l' append il puntatore del file si posizione dopo l' ultimo dato scritto pertanto non si perde nessun dato se lo si scrive, si apre con "a" prima il puntatore: FILE *Fp; fp= fopen("prova.txt" , "a");

Come scritto su il nome del file "prova.txt" questo file per trovarlo se cercarlo o se si crea viene scritto dentro la stessa cartella dove viene creato il file .exe se dobbiamo cercare o portarlo altrove si deve scrivere :
c:\\user\\prove\\prova.txt è possibile usare l' apertura del file in tutti i modi esempio

: FILE *Fp; fp= fopen("c:\\user\\prove\\prova.txt" , "r+" "w+" "a"); in questo modo il puntatore Fp si posiziona secondo chi trova per primo scritto se è un nuovo file da creare meglio prima w che inizia dall' inizio, se dobbiamo leggere il file è meglio r per primo comando , ma se dobbiamo continuare nello scrivere meglio inserire il comando a che inizia dalla fine del file nel nostro caso nell' esempio su Fp si posiziona all' inizio perché trova il comando leggi r

Per chiudere un file di si usa fclose per chiudere il nostro FILE *Fp : fclose (fp); dove fp è il nome prima del fopen fp=fopen("prova.---", "ab") ; ab è la formula per scrivere nel caso di un file binario

Ad ogni esecuzione di fopen si apre automaticamente dei standard (std)per comunicare con il file aperto stdin che è input , stdout che è output e stderr che sono errori

Per scrivere e leggere questi dati dentro un File non sono buoni printf e scanf ci sono fprintf e fscanf che scrivono proprio sui file, special modo scanf dopo un inserimento di uno spazio finisce con lo scrivere nell' input, mentre fscanf finisce quando si batte un invio da tastiera cosi possiamo avere stringhe un po' più lunghe di un semplice nome questo è un input e output formattato

Per scrivere i file di testo possiamo usare anche le stringhe con fgets e fputs , oppure possiamo usare la scrittura di caratteri con fgetc e fputc per sostituire fscanf e fprintf

Il file viene sempre letto fino alla fine finchè non trova i terminatori che possono essere \0 come per le stringhe oppure finisce con EOF (end of file) tutte le funzioni per i file i/o sono gestite dal file (stdio.h) file che si include all' inizio dei programmi di C

Prima di lasciare il C intendiamoci che il C non è solo queste poche cose che abbiamo visto c'è ne sono molte altre ma per noi ci può bastare giusto per collegarci con il C++ del nostro compilatore del VS2015 altre cose che troverò compatibili o che sono legate solo al C posso sempre trascriverle qui dentro il forum

Ciao grazie ByMpt-Zorobabele Di questo testo dalla prefazione fino all' 15° capitolo scritto, ho creato un testo PDF da scaricare lo trovate insieme dove rimane l' elenco delle foto e ogni volta sarà aggiornato con i nuovi testi se desiderate conoscere ancora di più il C potete proseguire nel leggere

Ecco dove io mi aggiorno dai seguenti testi o siti internet ;

http://www.cs.unibo.it/~ghini/didattica/sistemi1/c.html

http://lia.deis.unibo.it/Courses/FondT-0809-ELT/materiale/

http://www.diit.unict.it/users/michele/didattica/fondamenti/fondamenti/lingC.html

http://www.dsi.unifi.it/~poneti/

http://www.html.it/pag/15388/le-peculiarit-del-c/

http://www.mrwebmaster.it/c/guide/guida-c/

http://www.calderini.it/hycald/calderini_50_sist123/htm/pagineC.htm

https://it.wikibooks.org/wiki/C

https://msdn.microsoft.com/it-it/library/25db87se.aspx

testi manuali che ho scaricato

http://www.cerca-manuali.it/manuale-guida/c.htm