1. Cosa sono i puntatori?

I puntatori sono degli oggetti che hanno per valore l'indirizzo di memoria di un altro oggetto.

2. Come si creano?

I puntatori prendono il tipo in base al tipo dell'oggetto, quindi si inserisce prima il tipo, seguito dal simbolo '*' asterisco (in questo modo si viene a creare un tipo derivato) ed alla fine si sceglie l'identificatore della nostra variabile puntatore.

3.Come si usano?

Come prima cosa un puntatore deve essere inizializzato a NULL o con l'indirizzo di un oggetto, altrimenti avrà un indirizzo di memoria casuale. Si possono effettuare numerose azioni tramite i puntatori, con la dereferenziazione ad esempio si puo' accedere direttamente alla variabile puntata, in questo modo si puo modificare la variabile. Si possono utilizzare anche per mantenere l'indirizzo del primo elemento di un vettore o di una matrice, quindi si possono utilizzare per la gestione di vettori e matrici o anche di liste.

4.Quanto occupano in memoria?

Come gli interi, i puntatori occupano sempre 4 byte.

5.Possibili errori con puntatori non inizializzati a NULL?

Come abbiamo detto poco fa, un puntatore non inizializzato punta ad un indirizzo di memoria random.

6.Aritmetica dei puntatori

Per aritmetica di puntatori si intende tutta una serie di operazioni che si possono applicare sui puntatori. Sapendo che un puntatore ha in memoria l'indirizzo di memoria di un oggetto, se questo oggetto è ad esempio di tipo vettore, ed l'indirizzo del primo elemento è p, con l'espressione p+1 ritorna l'indirizzo di un oggetto di tipo vettore ma nell'area di memoria immediatamente successiva. Grazie ad un puntatore e l'aritmetica dei puntatori possiamo scorrere un intero vettore. Oppure, ad esempio se noi memorizziamo l'indirizzo del primo elemento di una matrice all'interno del nostro puntatore, con due cicli for uno per l'indice di riga e l'altro per l'indice di colonna, attraverso l'espressione (p+i*tot colonne +j) possiamo scorrere l'intera matrice. Questo tipo di matrice è detta matrice linearizzata.nt

7.Differenza tra puntatori a costanti (area di memoria assegnata) e puntatori costanti (indirizzo puntatore).

Ci sono due modi per utilizzare la parola chiave CONST con i puntatori. Nel primo caso la parola chiave CONST si inserisce prima del tipo del puntatore, sta a significare che con il puntatore non possiamo modificare l'oggetto da lui puntato, quindi l'oggetto a lui assegnato potrebbe essere anche una costante. Nel secondo caso, invece, la parola chiave CONST va inserita tra l'asterisco e l'identificatore, in questo modo sarà il puntatore un oggetto costante (come tutte le costanti del linguaggio) deve essere inizializzato e l'indirizzo a lui assegnato non potrà piu' essere cambiato.

- Int const *b = &a; //<-puntatore a intero costante
- Int *const b = &a; //<-puntatore costante a intero

8.Cos'è una lista di inizializzazione?

Una lista di inizializzazione non è altro che la prima parte di una funzione di un costruttore di classe. La funzione del costruttore è appunto divisa in due parti, lista di inizializzazione e corpo del costruttore. La lista di inizializzazione viene ad esistere se nella classe sono stati dichiarati dei membri dato costanti. La sintassi è : Dopo avere aver chiuso le parentesi tonde degli argomenti del costruttore si devono inserire i ':'due punti, subito dopo richiamare con il loro identificatore i membri dato costanti ed inizializzarli con un valore o con argomento formale della funzione costruttore. Nella lista di inizializzazione, le inizializzazioni dei membri dato costanti sono separati da una virgola.

9. Cos'è il costruttore di copia? e quello di default?

Il costruttore di copia è un particolare tipo di funzione che assume lo stesso nome della classe di appartenenza (come il costruttore) ed effettua una copia membro a membro di tutte le variabili già inizializzate dal costruttore, così da creare un oggetto identico a quello precedentemente creato. Il costruttore di default viene invocato automaticamente quando si ha la necessità, dal programma stesso, tuttavia è spesso necessario che il programmatore ridefinisca questo costruttore poiché i dati da copiare possono essere troppo complessi per quello di default.

10.Cos'è e come si gestisce la memoria dinamica?

La memoria dinamica o heap è una particolare tipo di memoria in cui gli indirizzi degli oggetti sono allocati in modo sparso, a differenza dello stack che sono allocati in modo sequenziale. La memoria dinamica viene utilizzando quando non si sa la quantità di memoria che un oggetto necessita a tempo di compilazione ma solamente a tempo di esecuzione. Per allocare oggetti in questa memoria si deve utilizzare la parola chiave NEW seguita dal tipo

dell'oggetto che si vuole allocare. L'operatore new restituisce l'indirizzo del blocco di memoria, per non perdere questo indirizzo lo si deve salvare in un puntatore creato in memoria stack. Se a questo puntatore viene assegnato un indirizzo di memoria diverso, l'oggetto in memoria dinamica precedentemente a lui assegnato viene perso definitivamente. L'oggetto continuerà ad esistere nella memoria dinamica fin quando non viene deallocato attraverso l'operatore delete, oppure automaticamente alla fine del programma.

11.Cos'è il puntatore this? A cosa serve? A cosa punta?

Il puntatore this è un particolare tipo di puntatore che viene dichiarato implicitamente nella classe. Questo puntatore prende l'indirizzo dell'istanza della classe che è stata creata. Questo puntatore puo' essere utilizzato solo dalle funzioni membro della classe stessa. Questo puntatore puo' essere utilizzato in svariati modi ad esempio quando si ha la necessita di avere un ritorno funzione dell'oggetto classe stesso o di una delle variabili del campo dati della classe o addirittura anche per fare un controllo dell'aliasing con un'altra istanza della medesima classe.

12.Qual'è la differenza tra parametri formali e attuali?

I parametri formali non sono altro che dei contenitori che vengono inizializzati con i parametri attuali passati alla funzione. Ad eccezione dei puntatori o al passaggio degli indirizzi, i parametri formali prendono il valore dei parametri attuali e non possono modificare i parametri attuali. Alla fine del blocco vengono distrutti. I parametri attuali sono invece variabili che vengono dichiarate fuori dai blocchi delle funzioni, ad esempio nel main. Vengono distrutti alla fine del programma.

13. Qual'è la differenza tra passaggio per valore e per riferimento?

Come abbiamo detto poco fa gli argomenti attuali possono essere passati alle funzioni: Tramite passaggio per valore, cioè vengono copiati esclusivamente i valori delle variabili negli argomenti formali, e la modifica di un argomento formale non si riperquote sull'argomento attuale a lui assegnato. Tramite passaggio per riferimento, cioè nell'argomento formale viene copiato l'indirizzo del blocco di memoria della variabile a lui assegnata, ed avremo un altro nome per quella varibile da usare all'interno della nostra funzione. Quest'ultimo tipo di passaggio si riperquote sull'argomento attuale in caso di modifica dell'argomento formale.

14.Cosa sono le stringhe?

Anche se implementate dalla libreria <string>, il tipo stringa non esiste in C++. Per far fronte a questa mancanza si è deciso di vedere le stringhe come array di caratteri, infatti, ogni cella contiene un carattere. Ogni stringa termina con un carattere di fine stringa che in questo caso è "\0" che mette lo stream in una situazione di errore(con precisione con l'errore viene settato ad 1 il flag di letture del carattere di fine stringa) interrompendo così le operazioni che si stanno effettuando su di essa. Possiamo effettuare svariate operazioni sulle stringhe includendo la libreria <cstring> come ad esempio: -La copia di una stringa in una variabile apposita. -Il confronto tra due stringhe. -Il controllo della lunghezza della stringa (escluso "\0")

15. Cos'è il Distruttore?

Il Distruttore è quella particolare funzione che viene richiamata automaticamente al termine del programma per deallocare tutta la memoria allocata durante l'esecuzione del programma.

16.Cosa sono le classi di memorizzazione?

Le classi di memorizzazione o storage class è una particolare proprieta' che definisce il tempo di vita di un oggetto all'interno del programma. Le classi di memorizzazione sono 3:

-Automatica: Sono gli oggetti che vengono dichiarati all'interno di un blocco. Vengono creati quando viene incontrata la loro definizione e vengono distrutti alla fine del blocco. Se non vengono inizializzati il loro valore è indefinito. Esempio: Oggetti formali di una funzione sono pensati come automatici.

-Statica: Sono gli oggetti che vengono dichiarati al di fuori di una funzione. Vengono creati/inizializzati all'inizio dell'esecuzione del programma principale e vengono distrutti alla fine del programma stesso. Se non sono stati inizializzati il loro valore è 0.

-Dinamica: Sono gli oggetti che vengono creati quando ad un certo punto del programma si incontra l'operatore NEW. Vengono distrutti o quando si incontra l'operatore DELETE o alla fine del programma stesso. Se non inizializzati hanno un valore indefinito.

17.Come si effetua la lettura/scrittura da/su di un file?

Per leggere o scrivere su di un file si utilizza la libreria <fstream> che contiene le operazioni di lettura e scrittura sui file. La lettura si effettua creando uno stream con la seguente sintassi: -ifstream nome ("nomefile.txt"); ed in seguito utilizzando il nome assegnato allo stream di lettura collegato al nostro file ed una variabile dello stesso tipo di quelle che dovrà essere letto dal file. es: int a =0;

nome>>a;

La scrittura si effettua creando uno stream con la seguente sintassi: -ofstream nome ("nomefile.txt"); ed in seguito utilizzando il nome del nostro stream di scrittura per scrivere cosa si vuole su file. es: char s= 'S';

nome<<s;

18.Cosa sono e quali sono gli algoritmi di ordinamento dei vettori?

Gli algoritmi di ordinamento di vettori sono dei procedimenti creati per riordinamento dei vettori in base alle condizioni richieste dalla situazione. I principali sono due:
-Bubble Sort: I valori piu grandi "Fluttuano" verso la parte finale dell'array quindi i valori passano da piu posizioni prima di raggiungere quella finale.
-Selection Sort: Ad ogni iterazione viene spostato un solo elemento (quello piu' piccolo) e lo mette all'inizio quindi in questo caso i valori arrivano direttamente alla loro posizione finale.

19.Cos'è il Define e a cosa serve?

In molti casi è utile assegnare a degli identificatori dei valori che restino costanti per tutto il programma e che non possono essere cambiati nemmeno per errore. In c++ questo si può ottenere in due modi: -Modificatore CONST -Direttiva DEFINE

Per quanto riguarda la direttiva DEFINE la sua sintassi è: Per la direttiva del preprocessore deve essere anteposto il simbolo del cancelletto (#) dopo la parola chiave DEFINE, in seguito l'identificatore e per conclusione il valore / operazioni. Ogni qual volta all'interno del programma si incontra l'identificatore del DEFINE, esso viene sostituito con il valore a lui associato. L'uso del DEFINE va oltre questi semplici collegamenti di indentificatori/valori ma può venire a costituire vere e proprie azioni che devono essere ripetute all'interno di un programma, le cosidette MACRO.

Una **MACRO** è appunto un operazione che deve essere svolta più volte in un programma Es -Se abbiamo necessita di uno scambio di variabili possiamo definire la seguente macro: #define scambia (x,y,temp) (temp)=(x); (x)=(y); (y)=(temp); a questo punto ogni qualvolta che viene incontrato la macro scambia viene sostituita alla serie di operazioni da compiere con quelle variabili.

20.Cos'è il cortocircuito AND e OR logico?

La valutazione a corto circuito è un meccanismo relativo agli operatori booleani binari, per cui il secondo operando viene valutato se e solo se il valore del primo operando non è sufficiente da solo a determinare il risultato dell'operatore. Il risultato dell'operatore logico AND è necessariamente falso se il primo operando è falso. Il risultato dell'operatore logico OR è necessariamente vero se il primo operando è vero. Quindi il risultato si può ottenere analizzando un solo operando (il primo).

Differenza tra left value e right value?

I nomi originariamente facevano semplicemente riferimento al lato *sinistro* ("l" per *left*) o al lato *destro* ("r" per *right*) di un'operazione di assegnazione, come ad esempio in

```
1. a = 6+2;
```

In questo caso sul lato *sinistro* c'è un *lvalue* a che è una variabile a cui è possibile assegnare un valore, mentre sul lato destro c'è un *rvalue* composto dall'espressione 6+2 alla quale non ha ovviamente senso poter assegnare un valore.
La distinzione nelle specifiche del C++ è importante in quanto si può ad esempio recuperare l'indirizzo in memoria di un *lvalue,* mentre non ha senso parlare di indirizzo di un *rvalue,* o di tentare di assegnare un nuovo valore ad un *rvalue*.

21.Differenza tra incremento Prefisso e Postfisso.

La differenza sostanziale tra incremento prefisso e postfisso sta nel momento dell'incremento della variabile e dal valore restituito. Nel prefisso incrementa prima la variabile da utilizzare e poi viene utilizzata dal programma (restituisce un left value). Nel postfisso il programma prima utilizza la variabile e poi viene incrementata (restituisce un right value). Dato che l'operatore di incremento può essere utilizzato su un lvalue è corretto effettuare un doppio incremento prefisso ma non un doppio incremento postfisso , appunto perche utilizzando il postfisso una volta viene restituito un rvalue su cui non puo' essere utilizzato un incremento.

22.Cosa e quali sono gli operatori logici.

Gli operatori logici sono dei particolari operatori che vengono utilizzati per effettuare dei controlli sulle variabili. I principali sono 3:

- **AND:** l'AND è un operatore logico binario che serve a verificare se due o più condizioni siano vere contemporaneamente. Si indica con la coppia di caratteri speciali "&&" (da non confondere con l'AND bit a bit "&").

- **OR**: l'OR è un operatore logico binario che serve a verificare se almeno una delle condizioni sia vera. Si indica con la coppia di caratteri speciali "||" (da non confondere con l'OR bit a bit "|").

- **NOT**: il NOT è un operatore logico unario, e serve a controllare che una condizione non si verifichi.

23. Cosa sono le liste?

Una lista è un tipo **derivato** composto da una struttura contenente uno o più membri informazione e uno o più membri puntatori. Le liste vengono utilizzate quando si ha la necessita di memorizzare un insieme di elementi ma senza saperne il numero a tempo di compilazione, creare un array in questo caso sarebbe stato errato per vari motivi esempio:

1. Per lo spreco di memoria allocando un array di dimensione arbitraria.

2. Per memoria insufficiente all'interno dell'array creato. Il concetto base delle liste è che dal primo elemento della lista è possibile ricondursi a tutti gli elementi successivi, appunto perché ai puntatori della struttura di tipo lista vengono assegnati gli indirizzi degli elementi di tipo lista successivi. E' possibile effettuare numerose operazioni sulle liste, eliminazione in coda/testa/ in base ad una condizione o inserimento in coda/testa/ in base ad una condizione o anche la stampa.

Le operazioni che si possono effettuare su una lista sono principalmente di due tipi: estrazione e aggiunta, rispettivamente in testa, coda e all'interno (a seconda di una condizione).

24. Cosa sono le funzioni friend?

Le funzioni friend sono particolari tipi di funzione che vengono dichiarate all'interno di una classe appunto come FRIEND "amiche" e possono accedere a tutti i membri privati e non di una classe, non facendo effettivamente parte della classe. Queste funzioni vengono poi definite al di fuori del file della classe.

25. Cos'è l'overloading?

L'Overloading (sovrapposizione) nel c++ puo' essere visto in due modi:

1. **Overloading delle funzioni**, cioè due funzioni con lo stesso identificatore possono esistere se e solo se almeno uno degli argomenti formali è di diverso tipo o se cambia il numero totale degli argomenti formali. ATTENZIONE: a differenza di altri linguaggi di programmazione, l'overloading di funzioni **non** può essere fatto, in c**++, con la sola differenza del tipo restituito**.

2. **Overloading degli operatori**, essendo le classi dei tipi derivati alcune operazioni non sono definite su di esse, quindi invece di definire una funzione che attua quel determinato procedimento il linguaggio ci permette di ridefinire l'operatore stesso. In questo modo ridefinendo il tipo degli operandi dell'operatore possiamo estendere ai tipi classe una proprietà già valida per i tipi fondamentali.

26. Cosa si intende per programmazione a moduli?

Per programmazione a moduli si intende un tipo di programmazione in cui il programma viene suddiviso in più file. Semplicemente:

- Modulo server (o servitore): quello che offre dei servizi.

- Modulo client: quello che sfrutta i servizi messi a disposizione del modulo server.

Questo metodo viene utilizzato per semplificare lo sviluppo, il test e la manutenzione del programma.

27.Cosa sono le matrici?

Le matrici o array multidimensionali sono una collezione di dati dello stesso tipo memorizzati sotto forma di matrice al quale si può accedere attraverso degli indici (uno rappresenta la riga l'altro la colonna) che individuano il valore all'interno di quella determinata posizione. Una matrice può essere vista anche come vettore linearizzato, in questo caso l'accesso ai vari membri della matrice avviene tramite l'espressione (riga corrente * numero colonne totale + colonna corrente).

28. Cos'è la grammatica di un linguaggio?

La grammatica di un linguaggio è l'insieme di regole che rappresentano il modo in cui bisogna usarlo. La grammatica è divisa in Sintassi e Semantica.

- **Sintassi**: Spiega le regole con cui vanno scritte le istruzioni.

- **Semantica**: Indica il loro significato.

29. Cosa sono le classi?

Le classi sono un tipo derivato che differiscono dalle strutture perché al loro interno i membri possono essere suddivisi in parte privata e parte pubblica. La parte privata può essere raggiunta dalle funzioni membro della classe dichiarate nella parte pubblica o dalle funzioni friend. Una classe è quindi un insieme di funzioni o variabili che trattano lo stesso argomento e che sfruttano gli stessi elementi per effettuare azioni diverse.

30. Cosa bisogna fare per impedire che un array venga modificato?
Basta aggiungere la parola chiave CONST prima del tipo dell'array.

31. Come si rappresentano i numeri reali? E cos'è il tipo enumerato?
I reali nel linguaggio c++ vengono rappresentati con la parola chiave DOUBLE, occupano 32 bit in memoria. Esistono delle varianti come il FLOAT che rappresenta sempre i reali ma su 16 bit ed il LONG DOUBLE su 64 bit. Sui reali sono definite tutte le operazioni aritmetiche e di confronto.

L'enumeratore è invece un tipo derivato che viene definito dal programmatore, quindi si viene a creare un nuovo tipo. Tutte le variabili con tipo enumeratore possono assumere solo i valori dichiarati dal tipo.

enumeratore. Es: enum giorni { lunedi, martedi, giovedi,ecc...}; -Le costanti dichiarate all'interno del enumeratore giorni vengono viste come costanti numeriche a partire da 0 per lunedi e 6 per domenica. -Creando una variabile di tipo giorni gli si puo' assegnare solo le costanti dichiarate all'interno dell'enumeratore giorni. -Il tipo Enumeratore viene utilizzato piu' che altro per le operazioni di confronto.

31. Cosa rappresenta il nome del puntatore?
Il nome del puntatore rappresenta l'identificativo che si riferisce all'indirizzo della cella di memoria occupata dalla variabile che ad esso è stata assegnata.

32. Cos'è il polimorfismo?
Il polimorfismo indica la possibilità di definire metodi e proprietà con lo stesso nome. Il polimorfismo puo' essere distinto in due casi: -Polimorfismo del contesto della programmazione ad oggetti:

Si riferisce al fatto che in una classe derivata possa ridefinire un metodo della classe base con lo stesso nome. Se per esempio è una funzione, questa deve essere preceduta dalla parola chiave VIRTUAL che sta a significare che è possibile avere un'altra definizione per la stessa funzione all'interno della classe derivata. La funzione all'interno della classe derivata deve essere preceduta dalla parola chiave OVERRIDE che sta a significare una sovrascrittura di un metodo che ha lo stesso nome del metodo della classe Base.

-Polimorfismo nel contesto della programmazione generica:

> Si riferisce al fatto che si può stilare un programma senza un tipo specifico. Questo nel c++ viene effettuato attraverso i Templates.

33. Operatori bit a bit.
1)AND bit a bit: (&) effettua un AND bit a bit tra due operandi, cioè, setta a 1 il bit quando i bit di quella posizione sono entrambi 1 e setta a 0 in tutti gli altri casi.

2)OR bit a bit: (|) effettua un OR bit a bit tra due operandi, cioè, setta a 1 il bit quando almeno uno dei bit in quella posizione è uguale ad 1 a 0 negli altri casi.

3)OR esclusivo bit a bit:(^)(caret) effettua un OR esclusivo bit a bit tra due operandi, cioè, setta

ad 1 il bit quando i bit in quella posizione sono diversi, in tutti gli altri casi setta il bit a 0.

Informatica (definizione informale): è la scienza della rappresentazione e dell'elaborazione dell'informazione Informatica (definizione formale dell'Association for Computing Machinery -

ACM): è lo studio sistematico degli algoritmi che descrivono e trasformano l'informazione, la loro teoria e analisi, il loro progetto, e la loro efficienza, realizzazione e applicazione.

Algoritmo: sequenza precisa (non ambigua) e finita di operazioni, che portano alla realizzazione di un compito. Le operazioni utilizzate appartengono ad una delle seguenti categorie: 1. Operazioni sequenziali Realizzano una singola azione. Quando l'azione è terminata passano all'operazione successiva. 2. Operazioni condizionali Controllano una condizione. In base al valore della condizione, selezionano l'operazione successiva da eseguire. 3. Operazioni iterative Ripetono l'esecuzione di un blocco di operazioni, finchè non è verificata una determinata condizione.

Calcolatori Elettronici come esecutori di algoritmi: gli algoritmi vengono descritti tramite programmi, cioè sequenze di istruzioni scritte in un opportuno linguaggio comprensibile al calcolatore.

Eseguibilità: ogni azione deve essere eseguibile dall'esecutore in un tempo finito Non-ambiguità: ogni azione deve essere univocamente interpretabile dall'esecutore Finitezza: il numero totale di azioni da eseguire, per ogni insieme di dati in ingresso, deve essere finito

Algoritmi equivalenti ∪ hanno lo stesso dominio di ingresso ∪ hanno lo stesso dominio di uscita ∪ in corrispondeza degli stessi valori del dominio di ingressso producono gli stessi valori del dominio di uscita

Proprietà essenziali degli algoritmi:

Correttezza: – un algoritmo è corretto se esso perviene alla soluzione del compito cui è preposto, senza difettare in alcun passo fondamentale (non ambiguità).

Efficienza: – un algoritmo è efficiente se perviene alla soluzione del compito cui è preposto nel modo più veloce possibile, compatibilmente con la sua correttezza.

Alfabeto V (lessico) - insieme dei simboli con cui si costruiscono le frasi Universo linguistico V* - insieme di tutte le frasi (sequenze finite) di elementi di V Linguaggio L su alfabeto V - un sottoinsieme di V* Come definire il sottoinsieme di V* che definisce il linguaggio? Specificando in modo preciso la sintassi delle frasi del linguaggio TRAMITE una grammatica formale

Grammatica G = V insieme finito di simboli terminali VN insieme finito di simboli non terminali P insieme finito di regole di produzione S simbolo non terminale detto simbolo iniziale Data una grammatica G, si dice Linguaggio LG generato da G l'insieme delle frasi di V - Derivabili dal simbolo iniziale S - Applicando le regole di produzione P Le frasi di un linguaggio di programmazione vengono dette programmi di tale linguaggio. Grammatiche 16 GRAMMATICA

Compilatore&Linker: traduce il programma sorgente in programma eseguibile

- ANALISI programma sorgente analisi lessicale analisi sintattica
- TRADUZIONE generazione del codice ottimizzazione del codice

Sviluppo di un programma (approccio interpretato):

- editing: scrivere il testo e memorizzarlo su supporti di memoria permanenti
- interpretazione
  - ANALISI programma sorgente analisi lessicale
  - analisi sintattica
  - ESECUZIONE

34.Cosa sono i litterals?

Sono costanti letterali, o litterals, sono tutte quelle codifiche che un programmatore immette da tastiera e rappresentano il valore che assume l'oggetto a cui è stato fatto l'assegnamento con il litteral. Es: int a = 5; //5 è un litteral che rappresenta il numero naturale 5

35 in c++ es di operazioni sequenziali, condizionali, iterative?

- Iterativa: cicli (while o for).
- Condizionale: switch e if.
- Sequenziale: tutte quelle non condizionali e non iterative.

36. cosa è la conversione implicita?

37. Cosa è un metalinguaggio?

E' un linguaggio che consente di descrivere sintassi e semantica di un ulteriore linguaggio. Per descrivere la sintassi del linguaggio c++ vengono adoperate notazioni non ambigue che possono essere spiegate mediante il linguaggio naturale. Per descriverne la semantica si ricorre direttamente al linguaggio naturale. La notazione adoperata è derivata direttamente dal classico formalismo Backus Naur Form.

Collegamento:

- un identificatore ha collegamento interno se si riferisce a una entità accessibile solo da quella unità di compilazione;
  - uno stesso identificatore che ha collegamento interno in più unità di compilazione si riferisce in ognuna a una entità diversa;
- in una unità di compilazione, un identificatore ha collegamento esterno se si riferisce a una entità accessibile anche ad altre unità di compilazione; – tale entità deve essere unica in tutto il programma.

Oggetto: • viene solo dichiarato se si usa la parola chiave extern (e se non viene specificato nessun valore iniziale); • viene anche definito se non viene usata la parola chiave extern (o se viene specificato un valore iniziale).