Struttura a blocchi che non e il main
Il main e dentro e poi ci sono altre scatolette che gli chiamiamo funzione 1,2,n che lavorano allo stesso livello di main. Per ora serve solo la struttura.
Il manin a sua volte può contenere dei blocchi, e i blocchi possono ottenere altri blocchi etc etc struttura di annidamento a blocchi. Per fortuna distinguere i blocchi e facile perche i blocchi sono i blocchi di ambito globale che e 1 tutto , i blocchi esterni ma in tutti contraddistinti da una stessa sintassi.e la sintassi dell main e funzioni e che è un tipo,nome della funzione dentro sono i parametri poi parentesi graffa aperta
 Type nomefunzione(parametri){
Corpo della funzione (e del main)
}
In generale un blocco, a esclusione dell'ambito globale che fa a se, un blocco che non sia una funzione, il globale.
 Cominciamo a parlare di dati.
 Vanno messi da qualche parte, a base livello come viene rappresentato , 0 e 1 . tutto quello che cera nel pc sono solo 0 e 1 che a livello hardware si concretizzano piu alta o piu basa altra 1 scambiano segnali elettrici.
 Tipi ; sono i domini degli oggetti (entità) che possiamo rappresentare, tra questi tipi base che sono predefiniti del linguaggio e sono “bool” valori booleani {vero,falso } in c++ true o false.
Per fare le domande o per formulare le domande il tipo boolean è fondamentale.
Int numeri interi con segno. 32bit
Poi ci sono i char caratteri.
\n va a capo
Float 32bit ,double 64 bit numeri reali,razionali, floating-point
Long long int,long int che possiamo rappresentare un po di piu anche per i double e float.
Floating point che si puo rappresentare con 128 bit.
Unsigned int negativi, signed int
Ha una intepretazione semantica diversa, bool diversa solo vero o falso se e 0 falso altro vero se intero , senza segno intero etc carattere etc

Per questo che il c++ si dice che sono tipati i dati che vengono utilizzati hanno tutti appiccicati un tipo e il tpo dice semplicemente che questo viene decodificato in questo modo
Altri tipi
Puntatori, verso la fine del corso (indirizzi)
Non base ci sono le stringhe (da std lib)sequenza di caratteri. Possono essere manipolati anche come dati
Costruttori di tipi noi vedremo solo i struct permettono di costruire tipi strutturati a partire di tipi piu semplici
 Vector (da std lib) sequenza di elementi di un tipo specificato
Struct e vector differenza, sequenza di oggetti stessa interi tanti interi tanit float booleano p etc , cose eterogenee struct,vector collezioni di oggetti tutti con la stessa struttura.

2 associare identificatori

 Variabili,costanti,memorizzare dari input,visualizzare dati (outpu)
costanti, cosa piu interessant varibaili,

Variabili sono contenitori di dati. Scatole che stano nella memoria di computer,mettere dati leggere dati, caratterizzate da un nome (“identificatore”) che e arbitrario, con dei limiti,può contenere solo caratteri alfanumerici e _ underscore non può cominciare per una cifra 0,9

Minuscole e maiuscole due cose diverse. Non può contenere spazzi
Da un tipo
A livello base in memoria vuol dire io nel codice specifico che ci sara una variabile che si chiama cosi posso scrivere nel codice int A; vuol dire da questo punto in poi (in questo blocco) tutte le volte che si fa riferimento a A si intende una variabile (scatoletta) che può contenere valori interi e che sta in un determinato posto in memoria.
Operazioni su una variabile
Solo due metterci valori e leggere valori. Oppure assegnare un nuovo valore.
A la variabile e una scatola che può contenere un solo valore di quell tipo li .
Il vuoto non esiste, in quella scatola c'è sempre qualcosa
Ogni volta che sovrascrivo, il contenuto precedente va perso.
Fonte di errori di programma
Non si consulta una variabile in lettura(non si va a guardare cosa c'è dentro il programma prima che il programma ci mette qualcosa).
Stare attenti a quello che facciamo
Come si fa a assegnare valori alle variabili.
Attraverso un linguaggio fondamentale che si chiama assegnazione che ha la sintassi seguente <variabile>= <espressione>
A=5+2;
pluto=”cane”;

pluto=”cane” + “” + “...” ;
+= string concatenazione
Espressioni

Definito prima e inizializzato x
(5*sqrt(2.0))/x , x deve essere una variabile o costante di tipo int/float/double etc,
sqrt=square root funzione di libreria di calcolo
In presenza di errori
Error- compilazione fallisce
Warning- compilazione va a buon fine con un avvertimento

Chat Gpt riassunto in ordine

mappa ; (informazioni in piu)

Assolutamente! Ho riorganizzato e integrato i tuoi appunti in modo più formale e strutturato, mantenendo i concetti originali e aggiungendo le precisazioni necessarie.

📝 Appunti Organizzati: Struttura del Programma e Dati

1. Struttura a Blocchi di un Programma (C++)

La struttura fondamentale di un programma è basata su blocchi di codice.

- Il blocco principale è chiamato **`main`**.
- Oltre al `main`, ci sono altre unità di codice chiamate **funzioni** (funzione 1, funzione 2, ..., funzione n), che operano allo stesso livello del `main`.
- Il **`main`** stesso può contenere al suo interno altri blocchi, e questi possono a loro volta contenerne degli altri, creando una **struttura di annidamento a blocchi** (o nidificazione).

📌 Sintassi dei Blocchi Esterni (Funzioni e `main`)

I blocchi esterni (`main` e le altre funzioni) sono contraddistinti da una sintassi specifica:

```
C++
Type nome_funzione(parametri){
   Corpo della funzione (e del main)
}
```

- **`Type`**: è il tipo di dato restituito dalla funzione (o dal `main`).
- **`nome_funzione`**: è l'identificatore della funzione (o `main`).
- **`parametri`**: sono i dati passati alla funzione per l'esecuzione (all'interno delle parentesi tonde).
- **`{}` (parentesi graffe)**: delimitano il **corpo della funzione** (il blocco di codice eseguibile).

📌 Ambito di Validità (Scope) dei Blocchi

La distinzione dei blocchi è cruciale per definire il loro **ambito di validità** (**scope**) (delle variabili e delle funzioni).

- **Ambito Globale**: Corrisponde a (tutto) il codice al di fuori del `main` e delle altre funzioni, (ed è unico).
- **Blocchi Interni**: Ogni blocco definito dalle parentesi graffe `{}` (che non sia la definizione di una funzione o l'ambito globale) delimita un ambito locale.

---

2. Rappresentazione dei Dati

A livello base (hardware), tutti i dati all'interno di un computer sono rappresentati tramite **segnali elettrici** che simboleggiano solo due stati: **0 e 1** (**bit**). (La differenza tra questi due stati è data da una tensione elettrica più alta o più bassa.)

Tipi di Dati (Type)

In C++, il linguaggio è **fortemente tipato**: ogni dato utilizzato deve avere un **tipo** associato, che ne definisce il **dominio** (l'insieme dei valori che può assumere) e stabilisce come la sequenza di **0 e 1** (in memoria) debba essere **decodificata** e interpretata (**interpretazione semantica**).

Tipi Base Predefiniti

Tipo	Descrizione	Dimensione (Tipica)	Esempi di Valore
`bool` (Boolean)	Valori booleani: **vero** o **falso**. Fondamentale per le **domande** o le **condizioni**.	(1 byte)	`true`, `false`
`char` (Character)	Singoli caratteri (lettere, numeri, simboli).	(1 byte)	`'a'`, `'B'`, `'\n'` (\n è il carattere di **a capo** - *newline*)
`int` (Integer)	Numeri interi con segno (positivi e negativi).	**32 bit**	5, −100
`float` (Floating-point)	Numeri reali/razionali in virgola mobile (precisione singola).	**32 bit**	3.14, −0.001
`double` (Double floating-point)	Numeri reali/razionali in virgola mobile (precisione doppia).	**64 bit**	3.14159265, 1.0E−5

Export to Sheets

Varianti e Modificatori:

- `long int` e `long long int`: interi con un **intervallo di valori** (più ampio) rispetto a `int` (tipicamente 64 bit per `long long int`).
- `long double`: floating-point con precisione estesa (fino a **128 bit**).
- `unsigned int`: interi **senza segno** (solo valori non negativi).
- `signed int`: interi **con segno** (questo è il default per `int`).

Altri Tipi Importanti

- **Stringhe (`std::string`)**: Una sequenza di caratteri. Non sono un tipo base, ma sono fornite dalla **libreria standard** (`std lib`). Possono essere manipolate come dati complessi.

- **Puntatori**: Tipi che contengono **indirizzi di memoria** (saranno trattati verso la fine del corso).

Costruttori di Tipi (Definizione di Tipi Strutturati)

- `struct` **(Structure)**: Permette di costruire tipi di dati **strutturati** a partire da tipi più semplici. Le `struct` contengono una collezione di oggetti **eterogenei** (oggetti di **diversi tipi**).
- `std::vector`: Una sequenza (dinamica) di elementi di un **tipo specificato**. È una collezione di oggetti **omogenei** (tutti con la **stessa struttura** e lo stesso tipo).

---

3. Identificatori: Variabili e Costanti

Variabili

Le **variabili** sono **contenitori di dati** (scatole concettuali) che risiedono nella **memoria** del computer. Servono per **memorizzare dati** (input) e **visualizzare dati** (output).

Caratteristiche di una Variabile

1. **Tipo**: Indica quale tipo di dato può contenere.
2. **Nome** (Identificatore): È un nome arbitrario, soggetto a regole:
    - Può contenere solo **caratteri alfanumerici** (a-z, A-Z, 0-9) e l'**underscore** (`_`).
    - **Non** può **cominciare** con una **cifra** (0-9).
    - È **case-sensitive** (le minuscole e le maiuscole sono considerate due cose diverse, e.g., `A` è diverso da `a`).
    - **Non** può contenere **spazi**.

Dichiarazione di una Variabile

Nel codice, la dichiarazione riserva spazio in memoria e stabilisce il tipo di dato che il contenitore può ospitare:

C++
int A; // Dichiara una variabile di nome A, che può contenere valori interi.

- Tutte le volte che, all'interno (di questo blocco), si fa riferimento a `A`, si intende il contenitore in un determinato posto in memoria.

Operazioni su una Variabile

Su una variabile sono permesse due operazioni fondamentali:

1. **Lettura**: Consultare il valore contenuto.
2. **Scrittura** (Assegnamento): Mettere (o sovrascrivere) un valore.

- Una variabile può contenere **un solo valore** del suo tipo alla volta.
- Il **vuoto non esiste** in una scatola (se non è inizializzata, contiene comunque un valore "sporco" o casuale - **garbage value**).
- Quando si **sovrascrive** (assegna un nuovo valore), il contenuto precedente **va perso**.

Assegnamento (Scrittura)

L'assegnamento è l'operazione fondamentale per assegnare valori alle variabili:

<variabile>=<espressione>

- Il **valore** dell'**espressione** a destra viene **calcolato** e poi **memorizzato** nella **variabile** a sinistra.

Esempi:

- `A = 5 + 2;`
- `pluto = "cane";`
- `pluto = "cane" + "" + "...";` (**concatenazione di stringhe**, spesso indicata con `+=` in C++).

Espressioni

Un'**espressione** è una combinazione di valori, variabili, operatori e chiamate di funzione che **produce un unico valore**.

Esempio:

- `(5 * sqrt(2.0)) / x`
  - **`x`** deve essere una variabile o costante di un tipo numerico (`int`, `float`, `double`, etc.) **precedentemente definita e inizializzata**.
  - **`sqrt`** (square root) è una **funzione di libreria** (di calcolo).

Gestione degli Errori (nel Compilatore)

- **Error (Errore)**: Il codice non è sintatticamente valido; la **compilazione fallisce**.
- **Warning (Avvertimento)**: Il codice presenta potenziali problemi o anomalie (e.g., una variabile dichiarata ma mai usata, o (il programma consulta) una variabile in lettura prima che (sia stato messo) un valore - **errore logico**); la compilazione va **a buon fine**, ma viene emesso un avvertimento.

Informazione Aggiunta

- (entità)
- (o nidificazione)
- (scope)
- (ed è unico)
- (di diversi tipi)
- (dinamica)
- (il programma consulta)
- (sia stato messo)
- (garbage value)
- (e.g.,)
- (di calcolo)
- (che è arbitrario)
- (della funzione)
- (e del main)
- (delle variabili e delle funzioni)
- (è una combinazione di valori, variabili, operatori e chiamate di funzione che produce un unico valore)
- (e.g., `A` è diverso da `a`)
- (concatenazione di stringhe)
- (di questo blocco)
- (o accidentale)
- (tipicamente 64 bit per `long long int`)
- (in C++ `true` o `false`)
- (o *newline*)
- (il codice non è sintatticamente valido)
- (il codice presenta potenziali problemi o anomalie)
- (o casuale)
- (La differenza tra questi due stati è data da una tensione elettrica più alta o più bassa)
- (di memoria)
- (all'interno delle parentesi tonde)
- (all'interno)
- (e l'underscore)
- (che ne definisce il dominio)
- (interpretazione semantica)
- (1 byte)
- (più ampio)
- (solo valori non negativi)
- (precisione singola)
- (precisione doppia)
- (che a livello hardware si concretizzano più alta o più bassa)