Gli algoritmi e i programmi

Mattia Cozzi
cozzimattia@gmail.com

a.s. 2023/2024

Contenuti

Modello del problema

Il modello del problema è una rappresentazione schematica di un particolare aspetto della realtà.

Vengono individuate:

- le entità, oggetti importanti ai fini della descrizione;
- le proprietà delle entità;
- le variabili;
- le costanti.

Modello del problema

Il modello del problema è una rappresentazione schematica di un particolare aspetto della realtà.

Vengono individuate:

- le entità, oggetti importanti ai fini della descrizione;
- le proprietà delle entità;
- le variabili;
- le costanti.

Modello del problema

Il modello del problema è una rappresentazione schematica di un particolare aspetto della realtà.

Vengono individuate:

- le entità, oggetti importanti ai fini della descrizione;
- le proprietà delle entità;
- le variabili;
- le costanti.

Modello del problema

Il modello del problema è una rappresentazione schematica di un particolare aspetto della realtà.

Vengono individuate:

- le entità, oggetti importanti ai fini della descrizione;
- le proprietà delle entità;
- le variabili;
- le costanti.

Modello del problema

Il modello del problema è una rappresentazione schematica di un particolare aspetto della realtà.

Vengono individuate:

- le entità, oggetti importanti ai fini della descrizione;
- le proprietà delle entità;
- le variabili;
- le costanti.

Dati e azioni

Distinguiamo:

- dati; valori assunti dalle variabili o dalle costanti;
- azioni, attività sui dati che permettono di ottenere il risultato.

I dati possono essere numerici, alfabetici o stringhe generiche.

Le azioni possono essere di tipo aritmetico o logico.

Dati e azioni

Distinguiamo:

- dati; valori assunti dalle variabili o dalle costanti;
- azioni, attività sui dati che permettono di ottenere il risultato.

I dati possono essere numerici, alfabetici o stringhe generiche.

Le azioni possono essere di tipo aritmetico o logico.

Dati e azioni

Distinguiamo:

- dati; valori assunti dalle variabili o dalle costanti;
- azioni, attività sui dati che permettono di ottenere il risultato.

I dati possono essere numerici, alfabetici o stringhe generiche.

Le azioni possono essere di tipo aritmetico o logico.

Dati e azioni

Distinguiamo:

- dati; valori assunti dalle variabili o dalle costanti;
- azioni, attività sui dati che permettono di ottenere il risultato.

I dati possono essere numerici, alfabetici o stringhe generiche.

Le azioni possono essere di tipo aritmetico o logico.

Procedimenti per la risoluzione di un problema

1. Descrizione del problema:
   - individuazione dei dati di input;
   - individuazione dei dati di output;
   - individuazione delle risorse a disposizione.

2. Stesura dell'algoritmo:
   - scrittura con diagrammi;
   - implementazione;
   - controllo e debug.

Procedimenti per la risoluzione di un problema

1. Descrizione del problema:
   - individuazione dei dati di input;
   - individuazione dei dati di output;
   - individuazione delle risorse a disposizione.
2. Stesura dell'algoritmo:
   - scrittura con diagrammi;
   - implementazione;
   - controllo e debug.

Procedimenti per la risoluzione di un problema

1. Descrizione del problema:
   - individuazione dei dati di input;
   - individuazione dei dati di output;
   - individuazione delle risorse a disposizione.

2. Stesura dell'algoritmo:
   - scrittura con diagrammi;
   - implementazione;
   - controllo e debug.

Procedimenti per la risoluzione di un problema

1. Descrizione del problema:
   - individuazione dei dati di input;
   - individuazione dei dati di output;
   - individuazione delle risorse a disposizione.

2. Stesura dell'algoritmo:
   - scrittura con diagrammi;
   - implementazione;
   - controllo e debug.

Procedimenti per la risoluzione di un problema

1. Descrizione del problema:
   - individuazione dei dati di input;
   - individuazione dei dati di output;
   - individuazione delle risorse a disposizione.

2. Stesura dell'algoritmo:
   - scrittura con diagrammi;
   - implementazione;
   - controllo e debug.

Procedimenti per la risoluzione di un problema

1. Descrizione del problema:
   - individuazione dei dati di input;
   - individuazione dei dati di output;
   - individuazione delle risorse a disposizione.

2. Stesura dell'algoritmo:
   - scrittura con diagrammi;
   - implementazione;
   - controllo e debug.

Procedimenti per la risoluzione di un problema

1. Descrizione del problema:
   - individuazione dei dati di input;
   - individuazione dei dati di output;
   - individuazione delle risorse a disposizione.

2. Stesura dell'algoritmo:
   - scrittura con diagrammi;
   - implementazione;
   - controllo e debug.

Procedimenti per la risoluzione di un problema

1. Descrizione del problema:
   - individuazione dei dati di input;
   - individuazione dei dati di output;
   - individuazione delle risorse a disposizione.

2. Stesura dell'algoritmo:
   - scrittura con diagrammi;
   - implementazione;
   - controllo e debug.

Definizione di algoritmo

Il termine viene dall'algebrista persiano del IX secolo al-Khuwarizmi.

Definizione

Un algoritmo è la descrizione di un insieme finito di istruzioni che devono essere eseguite per portare a termine un dato compito.

Esempi: istruzioni di montaggio di un mobile, ricetta di cucina, somme in colonna.

Ogni algoritmo prevede la presenza di un esecutore.

Definizione di algoritmo

Il termine viene dall'algebrista persiano del IX secolo al-Khuwarizmi.

Definizione

Un algoritmo è la descrizione di un insieme finito di istruzioni che devono essere eseguite per portare a termine un dato compito.

Esempi: istruzioni di montaggio di un mobile, ricetta di cucina, somme in colonna.

Ogni algoritmo prevede la presenza di un esecutore.

Definizione di algoritmo

Il termine viene dall'algebrista persiano del IX secolo al-Khuwarizmi.

Definizione

Un algoritmo è la descrizione di un insieme finito di istruzioni che devono essere eseguite per portare a termine un dato compito.

Esempi: istruzioni di montaggio di un mobile, ricetta di cucina, somme in colonna.

Ogni algoritmo prevede la presenza di un esecutore.

Definizione di algoritmo

Il termine viene dall'algebrista persiano del IX secolo al-Khuwarizmi.

Definizione

Un algoritmo è la descrizione di un insieme finito di istruzioni che devono essere eseguite per portare a termine un dato compito.

Esempi: istruzioni di montaggio di un mobile, ricetta di cucina, somme in colonna.

Ogni algoritmo prevede la presenza di un esecutore.

Pseudocodifica

È la descrizione di un algoritmo utilizzando il linguaggio comune secondo una serie di regole rigorose e un vocabolario ristretto.

Caratteristiche:

- Un algoritmo viene aperto e chiuso dalle parole `inizio` e `fine`.
- Operazioni di *input*: `immetti`, `leggi`, `acquisisci`, `read`.
- Operazioni di *output*: `scrivi`, `mostra`, `comunica`, `write`.

Pseudocodifica

È la descrizione di un algoritmo utilizzando il linguaggio comune secondo una serie di regole rigorose e un vocabolario ristretto.

Caratteristiche:

- Un algoritmo viene aperto e chiuso dalle parole `inizio` e `fine`.
- Operazioni di *input*: `immetti`, `leggi`, `acquisisci`, `read`.
- Operazioni di *output*: `scrivi`, `mostra`, `comunica`, `write`.

Pseudocodifica

È la descrizione di un algoritmo utilizzando il linguaggio comune secondo una serie di regole rigorose e un vocabolario ristretto.

Caratteristiche:

- Un algoritmo viene aperto e chiuso dalle parole `inizio` e `fine`.
- Operazioni di *input*: `immetti`, `leggi`, `acquisisci`, `read`.
- Operazioni di *output*: `scrivi`, `mostra`, `comunica`, `write`.

Operatori

Nel linguaggio di pseudocodifica possiamo utilizzare diversi operatori:

- assegnazione di un valore ad una variabile:

```
assegna x = 9;
calcola y = x + 3;
```

- operatori matematici;
- operatori di confronto;
- operatori logici (`AND`, `OR`, `NOT`, `XOR`).

Operatori

Nel linguaggio di pseudocodifica possiamo utilizzare diversi operatori:

- assegnazione di un valore ad una variabile:

```
assegna x = 9;
calcola y = x + 3;
```

- operatori matematici;
- operatori di confronto;
- operatori logici (`AND`, `OR`, `NOT`, `XOR`).

Operatori

Nel linguaggio di pseudocodifica possiamo utilizzare diversi operatori:

- assegnazione di un valore ad una variabile:

```
assegna x = 9;
calcola y = x + 3;
```

- operatori matematici;
- operatori di confronto;
- operatori logici (`AND`, `OR`, `NOT`, `XOR`).

Operatori

Nel linguaggio di pseudocodifica possiamo utilizzare diversi operatori:

- assegnazione di un valore ad una variabile:

```
assegna x = 9;
calcola y = x + 3;
```

- operatori matematici;
- operatori di confronto;
- operatori logici (`AND`, `OR`, `NOT`, `XOR`).

Parole chiave

In pseudocodifica si usano alcune parole speciali che permettono di strutturare logicamente l'algoritmo.

- `se`;
- `allora`;
- `altrimenti`;
- `fine se`;
- `esegui`;
- `finché`;
- `mentre`;
- `ripeti`.

Esempio 1

Algoritmo in pseudocodifica per il calcolo dell'area di un triangolo:

```
inizio
   immetti base;
   immetti altezza;
   calcola area = .5 * base * altezza;
   scrivi area
fine
```

Esempio 2

Algoritmo in pseudocodifica per salutare in base all'ora del giorno:

```
inizio
   acquisisci ora;
   se ora < 12:00:
   allora
      scrivi 'Buongiorno';
   se ora < 18:00:
   allora
      scrivi 'Buon pomeriggio';
   altrimenti:
      scrivi 'Buonasera';
   fine se;
fine
```

Diagrammi a blocchi (*flowchart*)

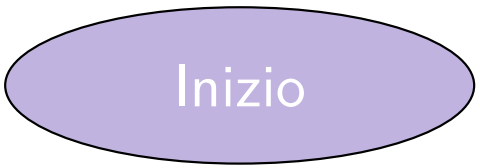
I diagrammi a blocchi permettono di rappresentare graficamente l'algoritmo.

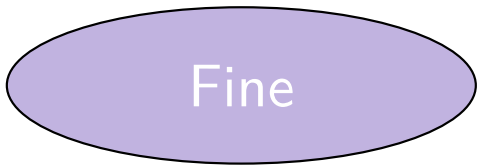
In questi schemi, blocchi di forme diverse hanno significati diversi.

Diagrammi a blocchi (*flowchart*)

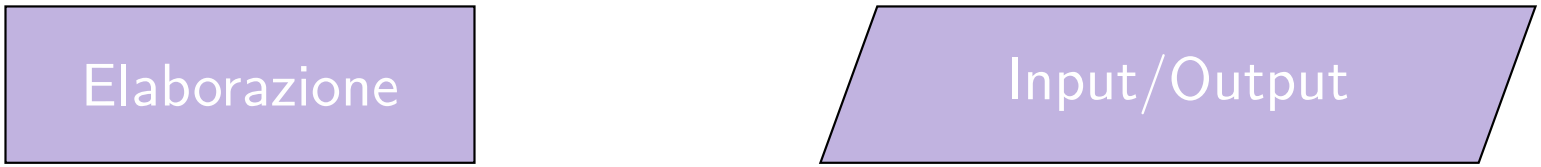
I diagrammi a blocchi permettono di rappresentare graficamente l'algoritmo.

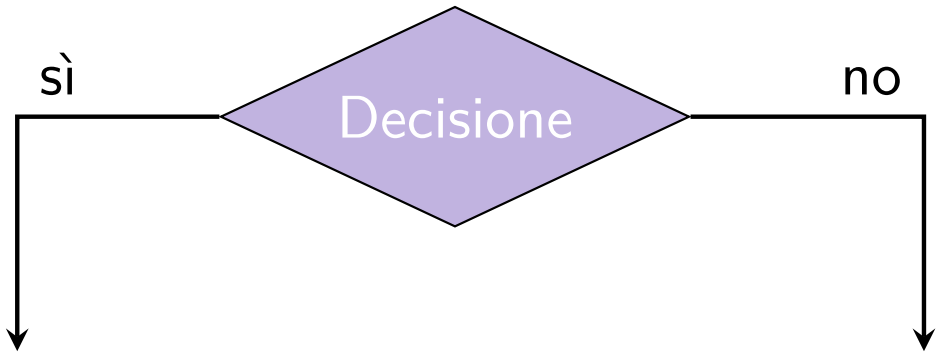
In questi schemi, blocchi di forme diverse hanno significati diversi.



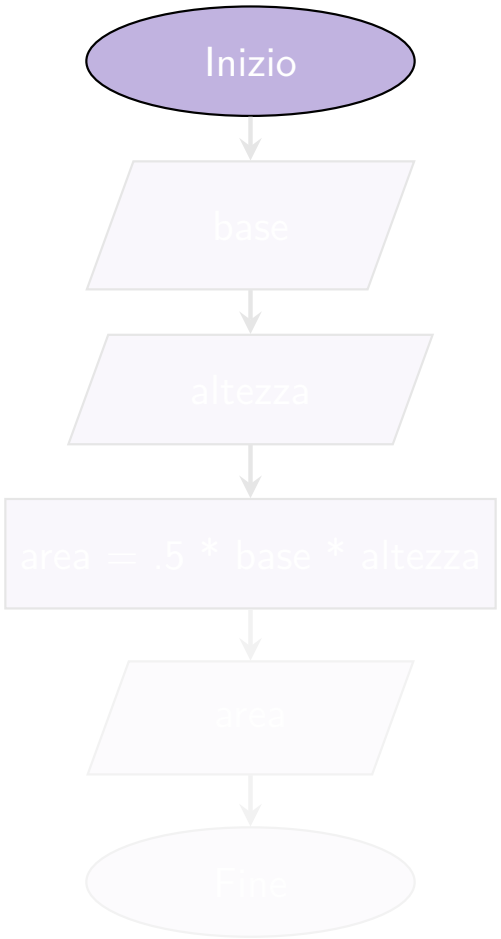


Tipi di blocchi

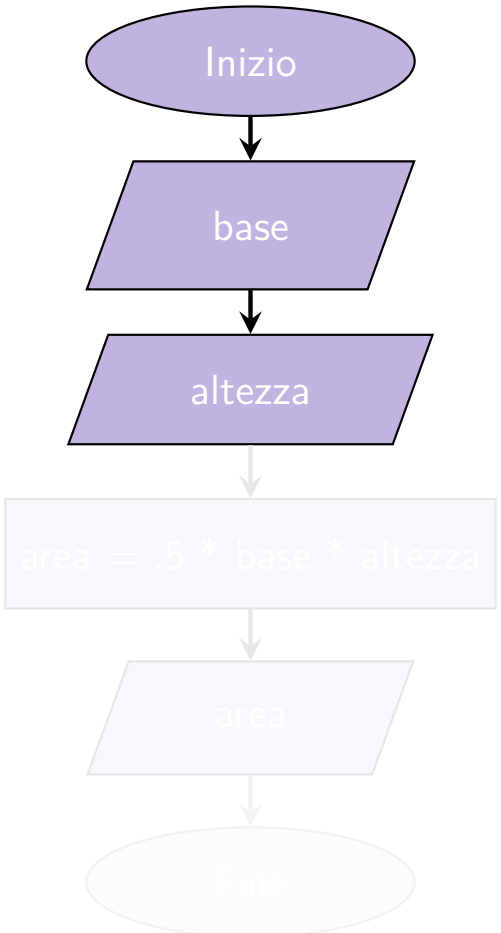




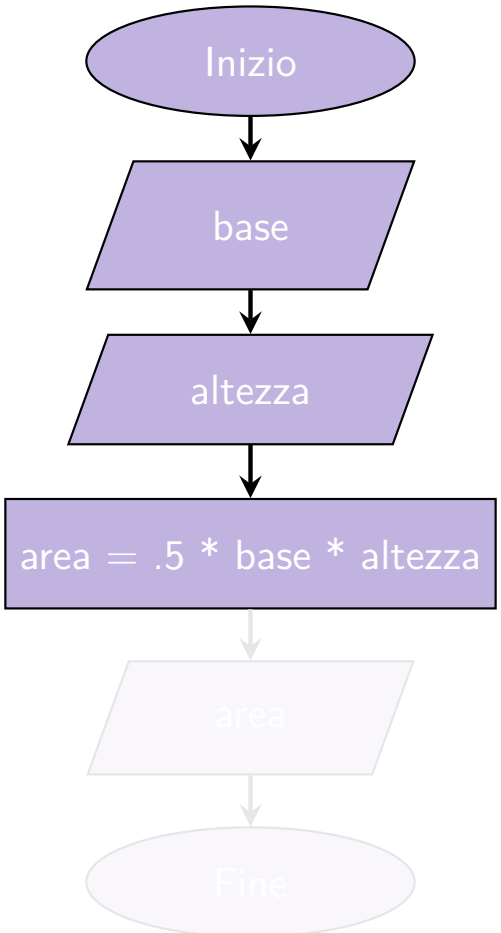
Esempio 1 a blocchi



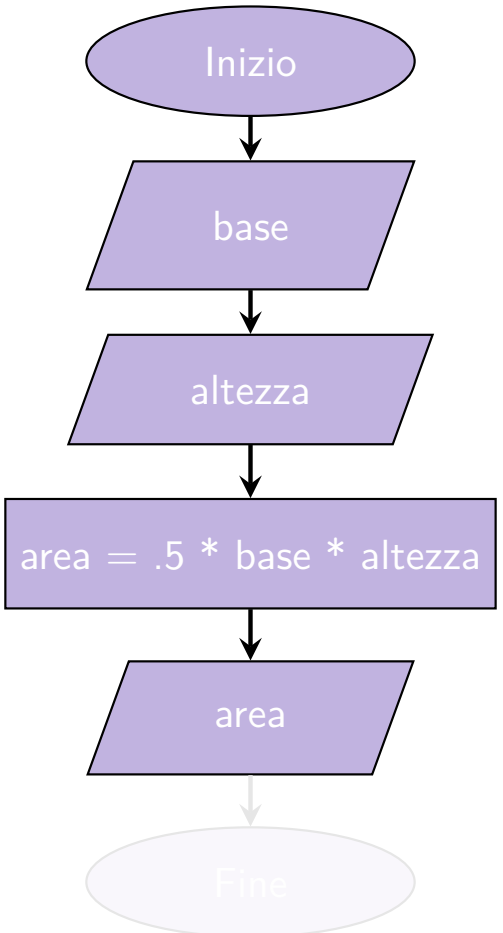
Esempio 1 a blocchi



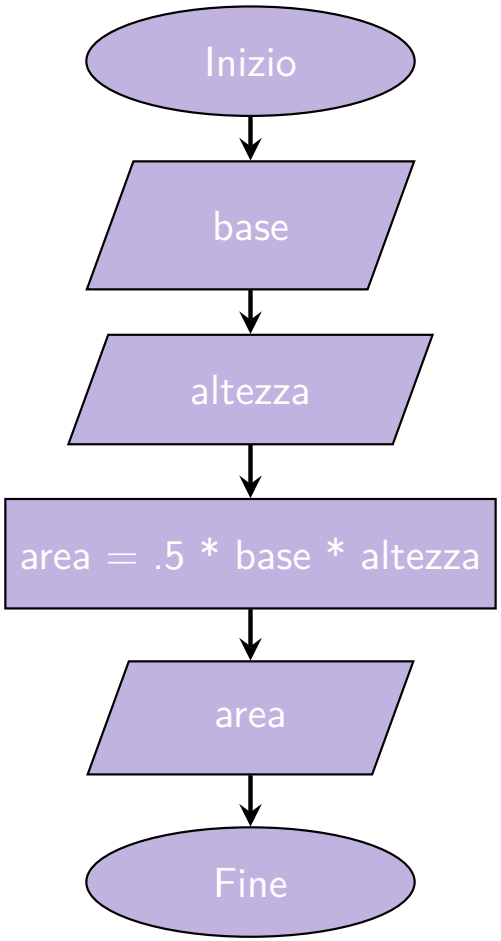
Esempio 1 a blocchi



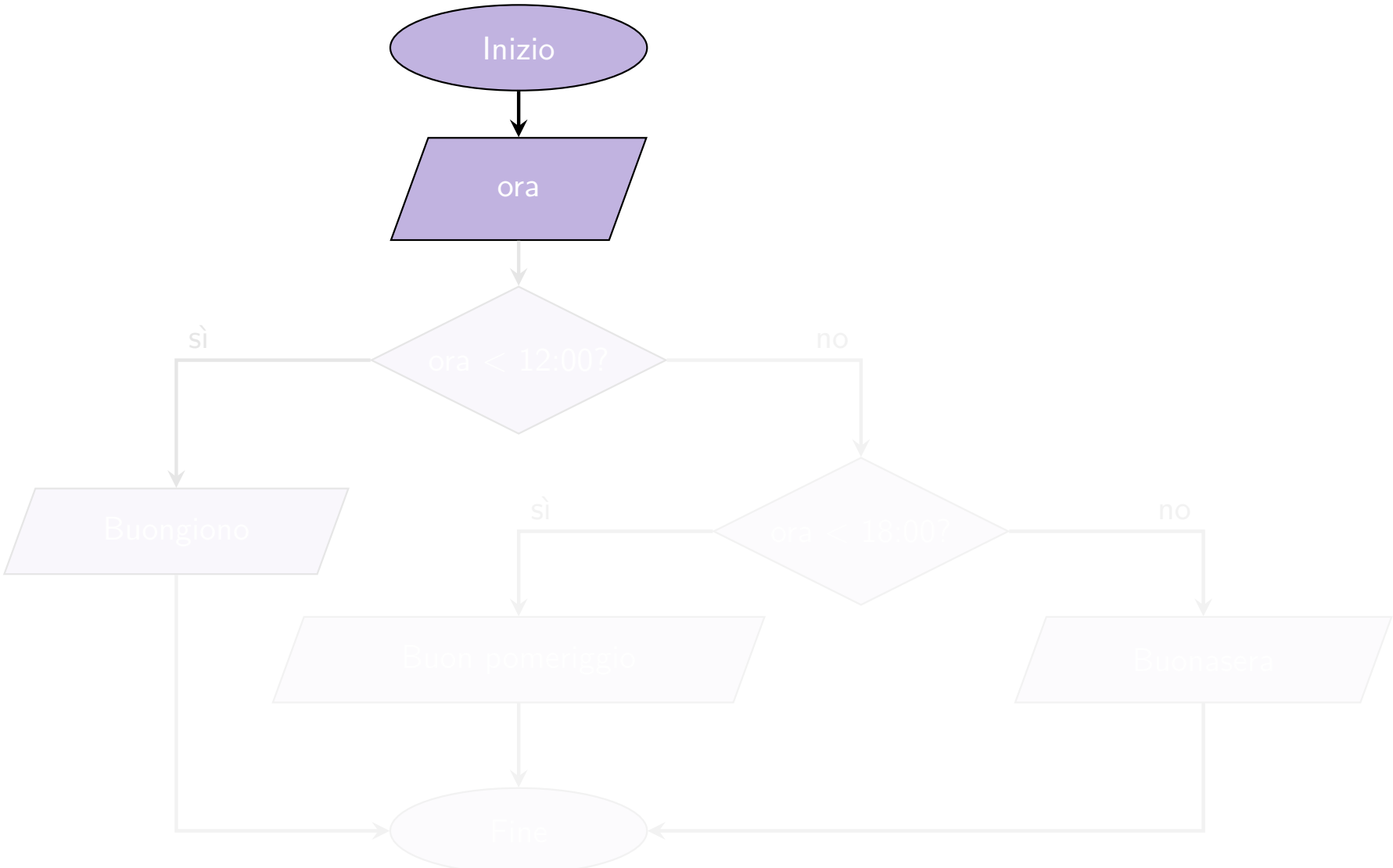
Esempio 1 a blocchi



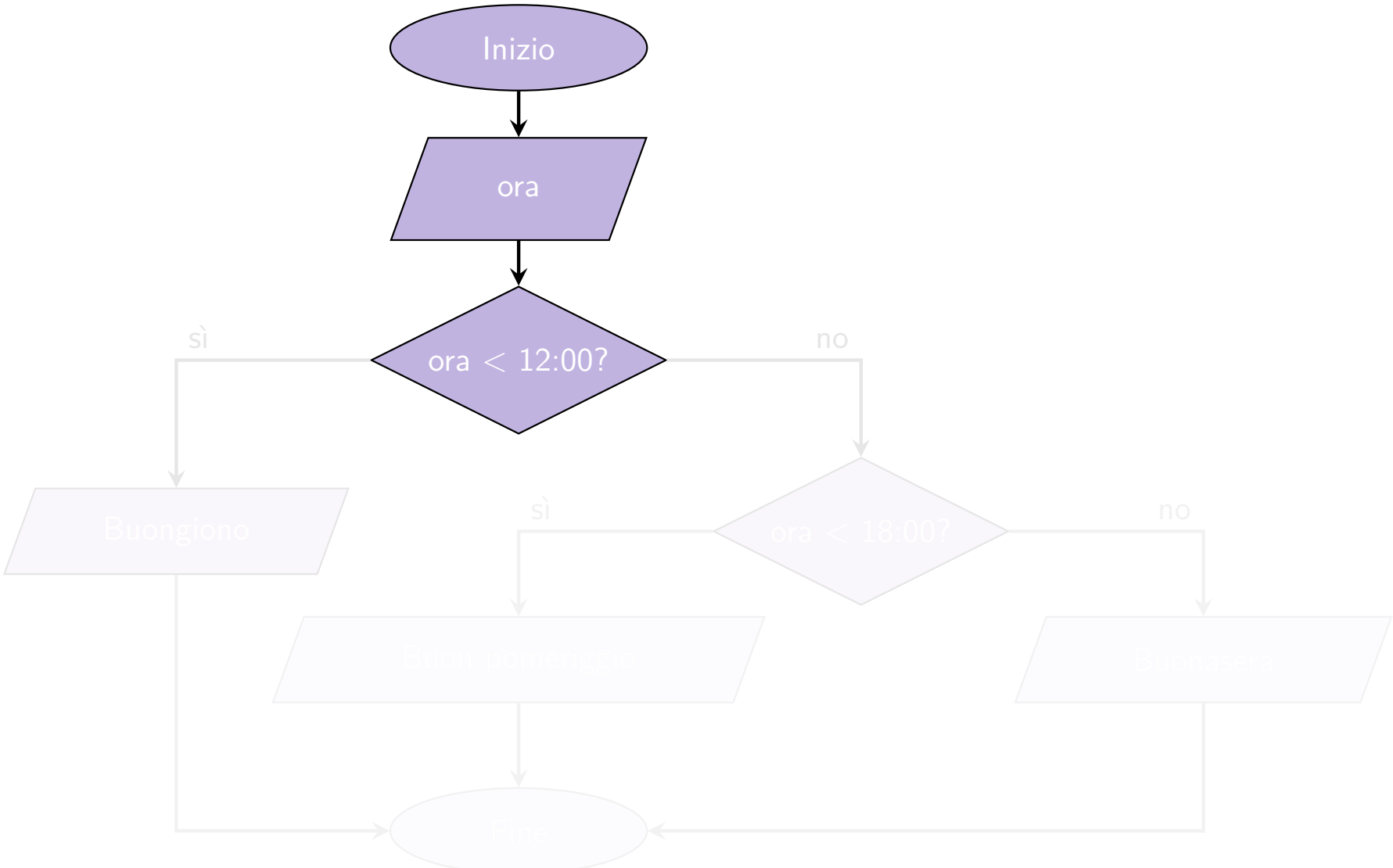
Esempio 1 a blocchi



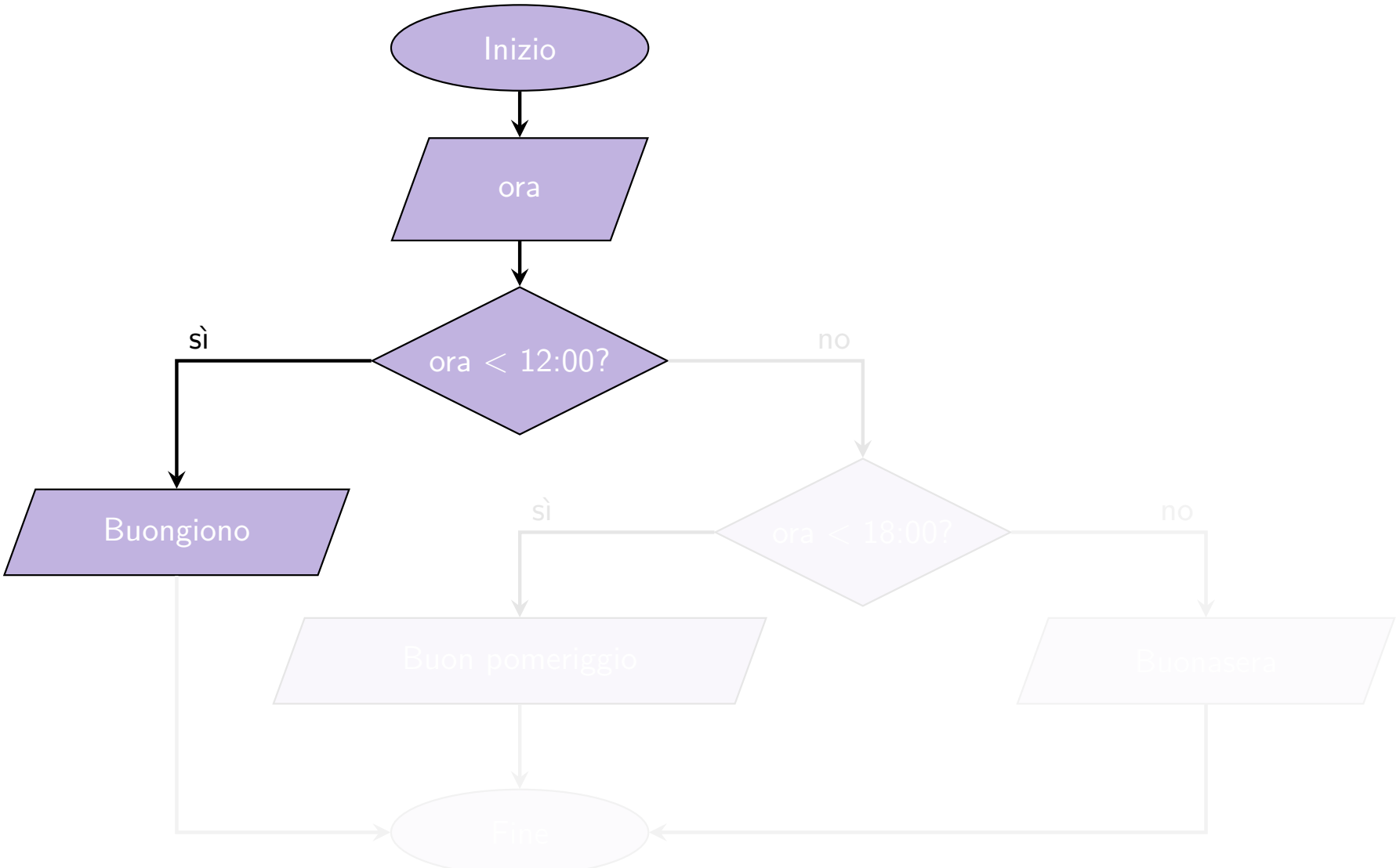
Esempio 2 a blocchi



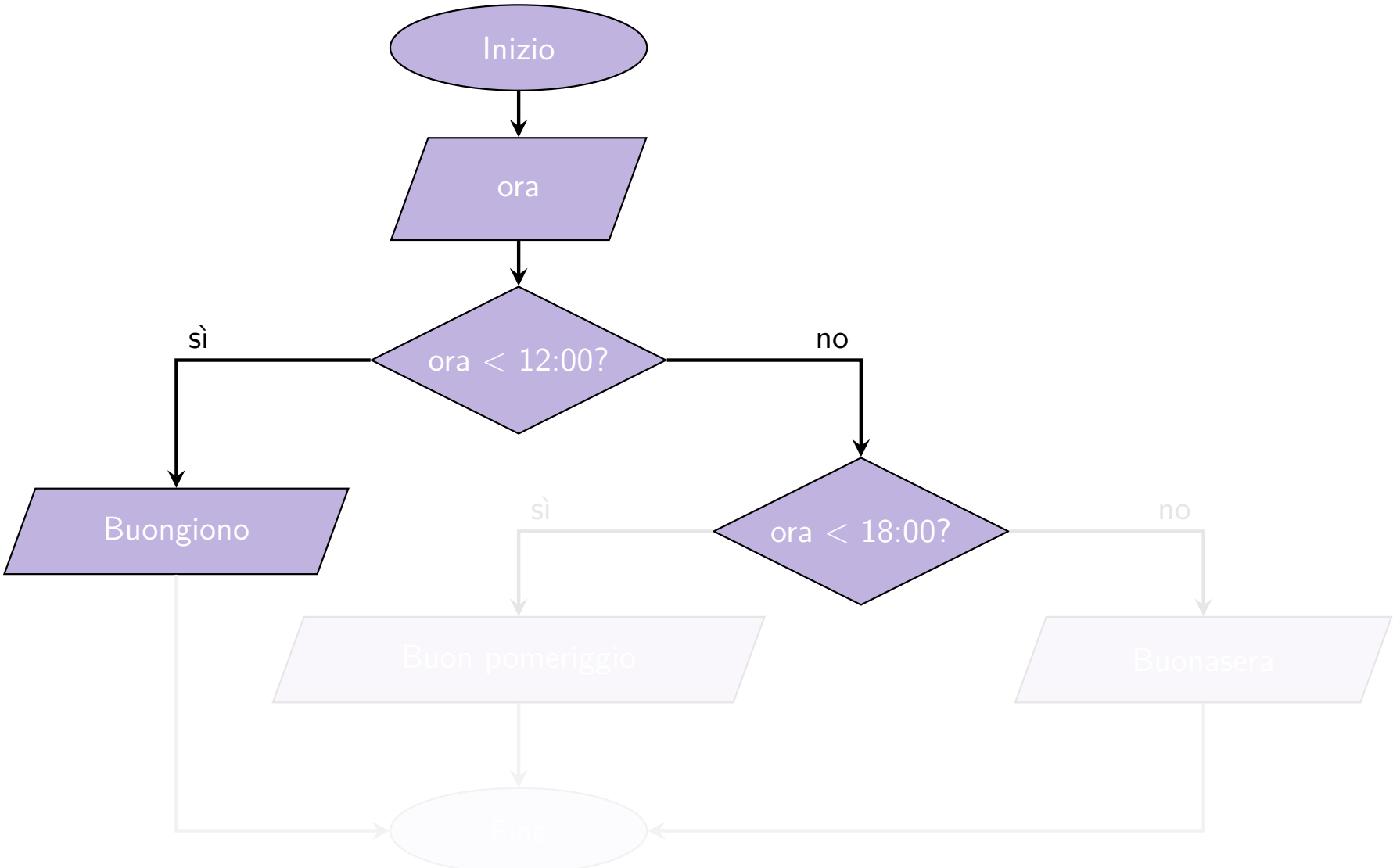
Esempio 2 a blocchi



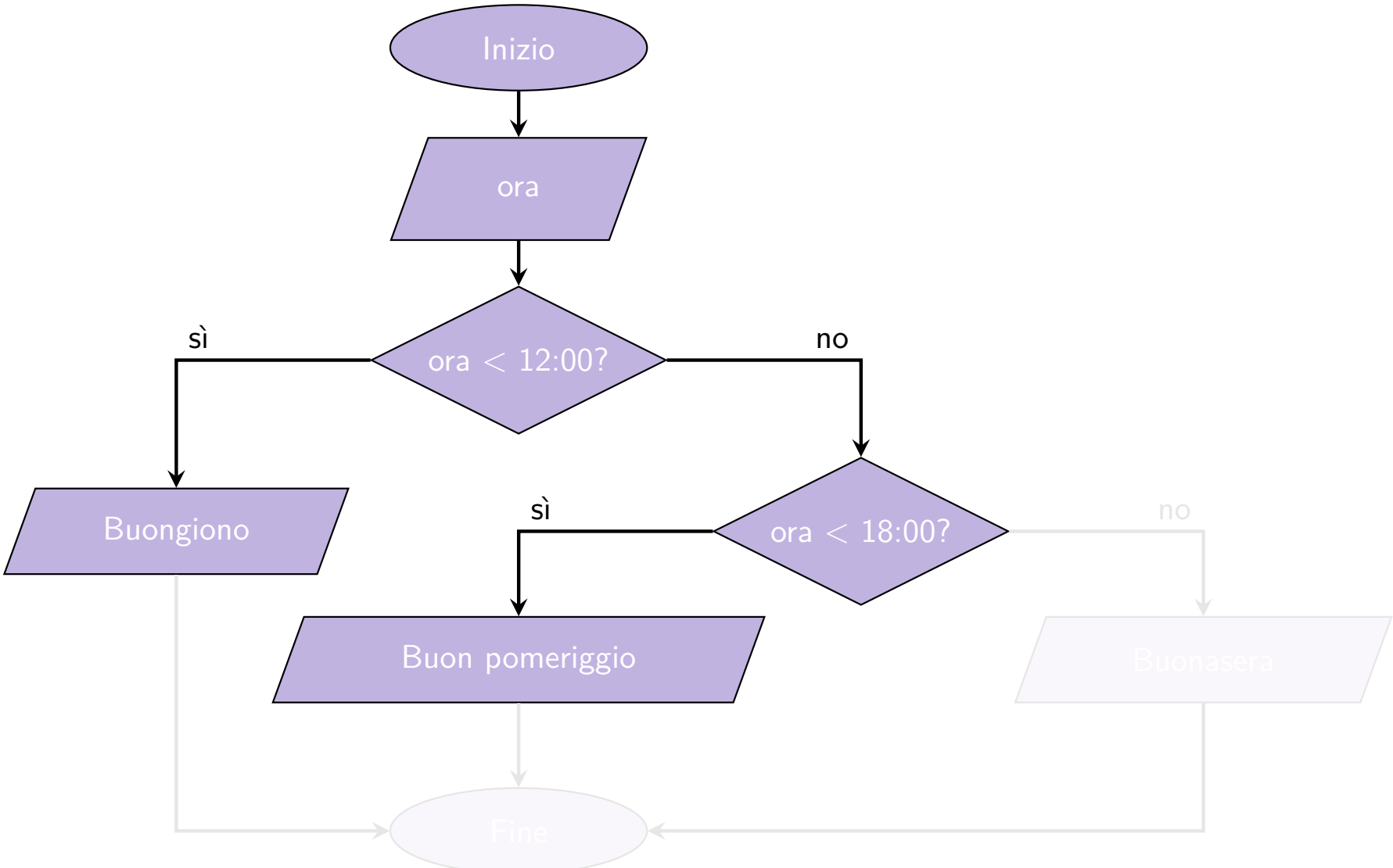
Esempio 2 a blocchi



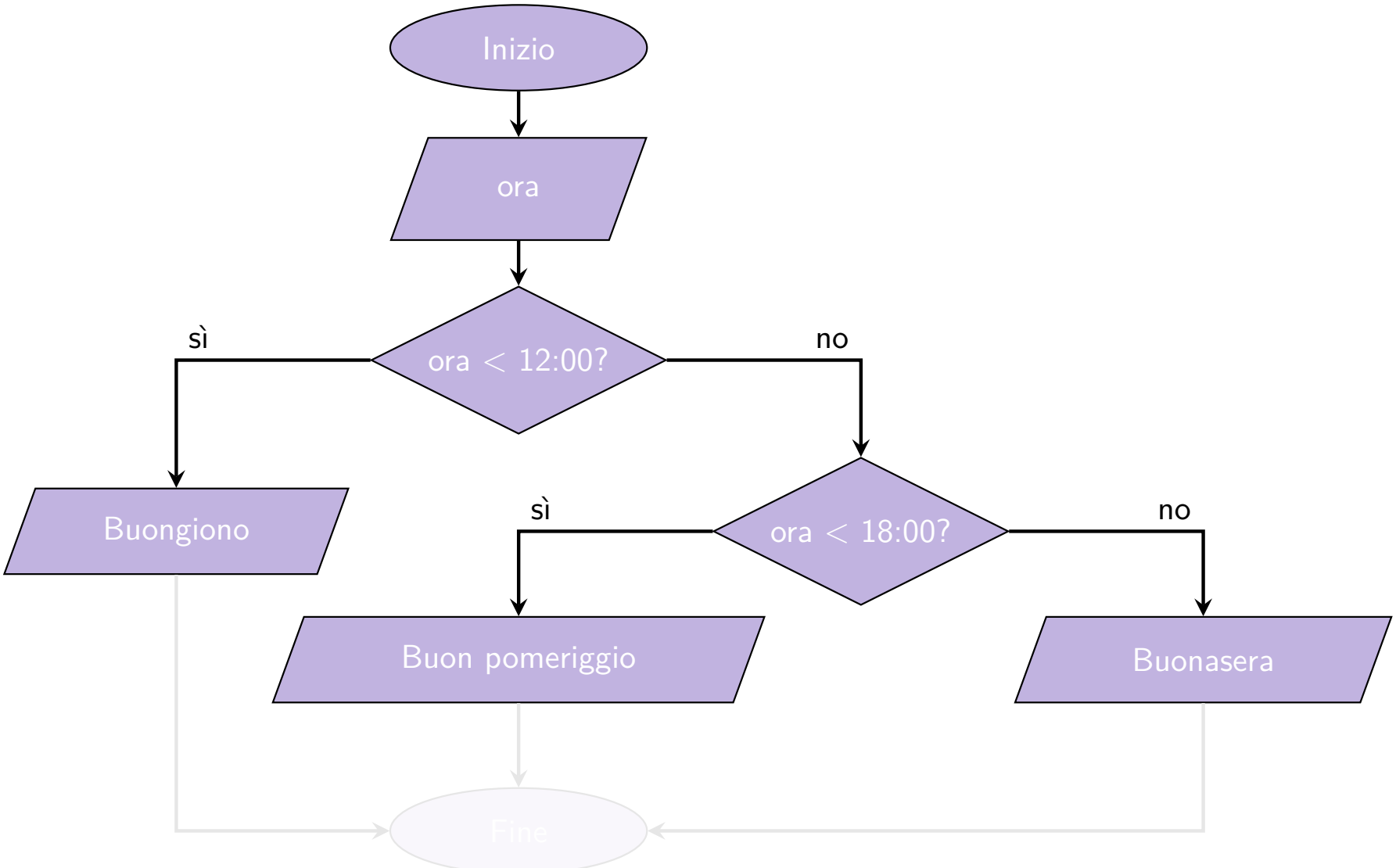
Esempio 2 a blocchi



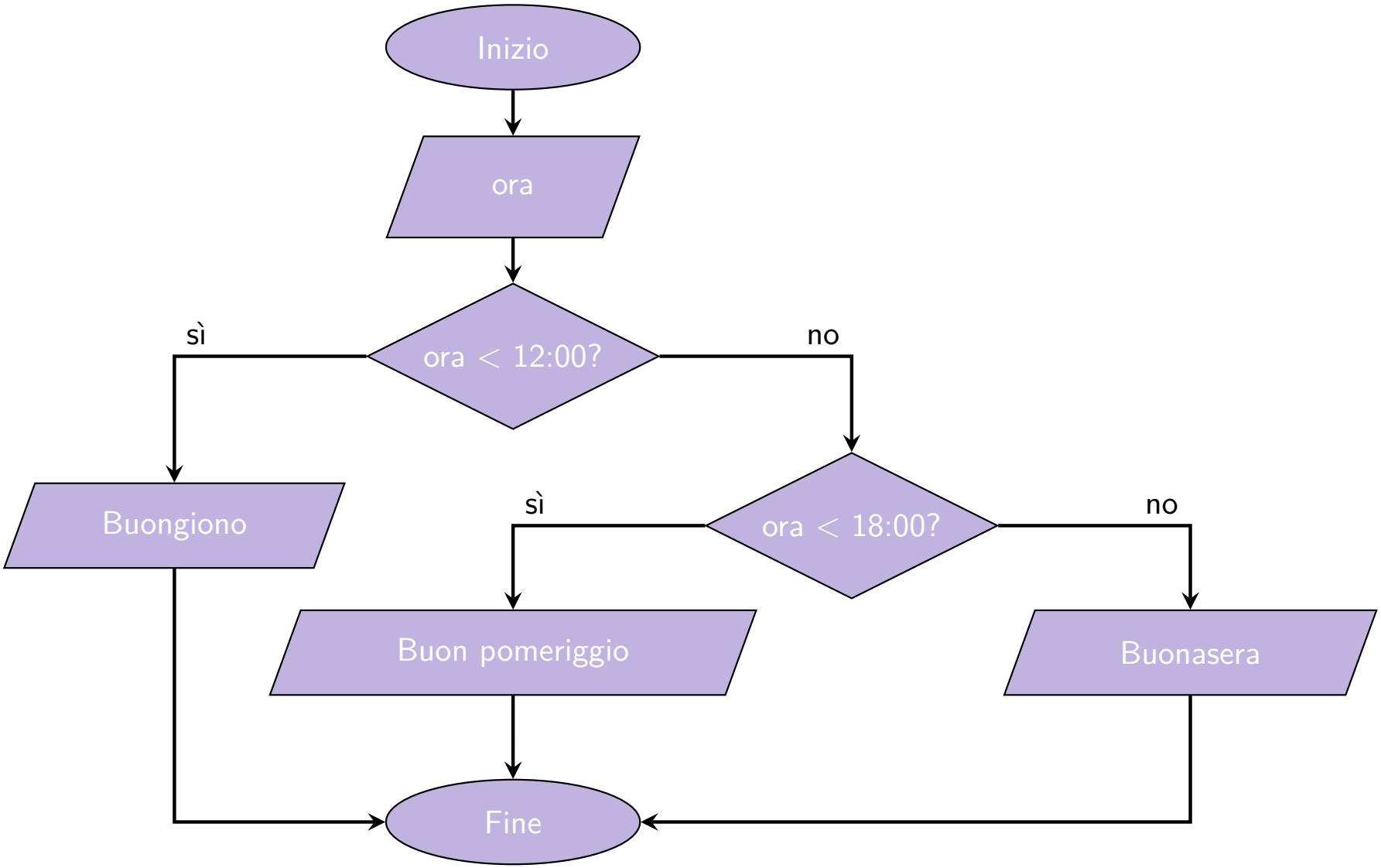
Esempio 2 a blocchi



Esempio 2 a blocchi



Esempio 2 a blocchi



Strutture di controllo

Le istruzioni di un algoritmo possono:

- essere organizzate in sequenza;
- presentare delle alternative (struttura condizionale);
- essere ripetute un certo numero di volte o finché si verifica una certa condizione (struttura iterativa).

Ogni algoritmo può essere scritto con una combinazione di queste tre strutture fondamentali.

Strutture di controllo

Le istruzioni di un algoritmo possono:

- essere organizzate in sequenza;
- presentare delle alternative (struttura condizionale);
- essere ripetute un certo numero di volte o finché si verifica una certa condizione (struttura iterativa).

Ogni algoritmo può essere scritto con una combinazione di queste tre strutture fondamentali.

Strutture di controllo

Le istruzioni di un algoritmo possono:

- essere organizzate in sequenza;
- presentare delle alternative (struttura condizionale);
- essere ripetute un certo numero di volte o finché si verifica una certa condizione (struttura iterativa).

Ogni algoritmo può essere scritto con una combinazione di queste tre strutture fondamentali.

Strutture di controllo

Le istruzioni di un algoritmo possono:

- essere organizzate in sequenza;
- presentare delle alternative (struttura condizionale);
- essere ripetute un certo numero di volte o finché si verifica una certa condizione (struttura iterativa).

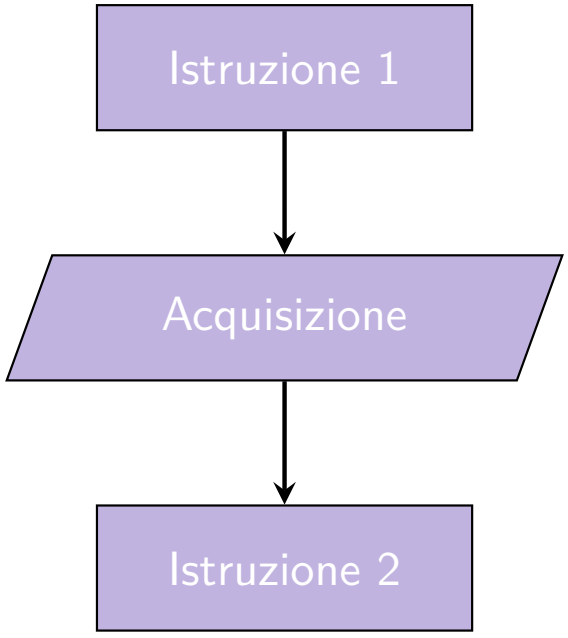
Ogni algoritmo può essere scritto con una combinazione di queste tre strutture fondamentali.

Sequenza

Pseudocodifica

```
Istruzione 1
Acquisizione 1
Istruzione 2
```

Blocchi

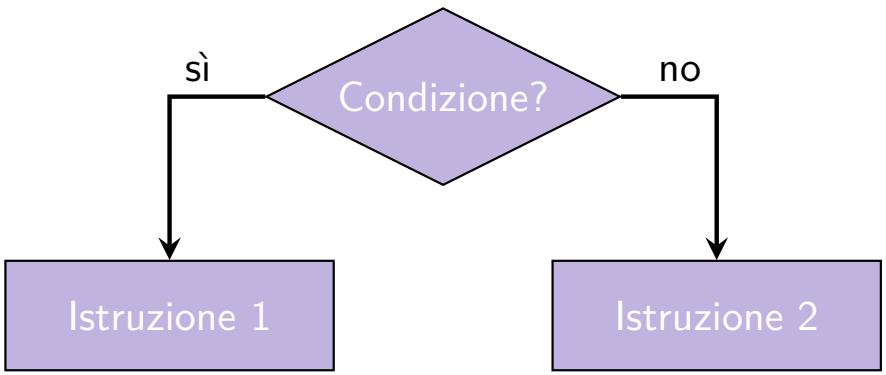


Struttura condizionale

Pseudocodifica

```
se condizione
allora
   istruzione 1
altrimenti
   istruzione 2
fine se
```

Blocchi

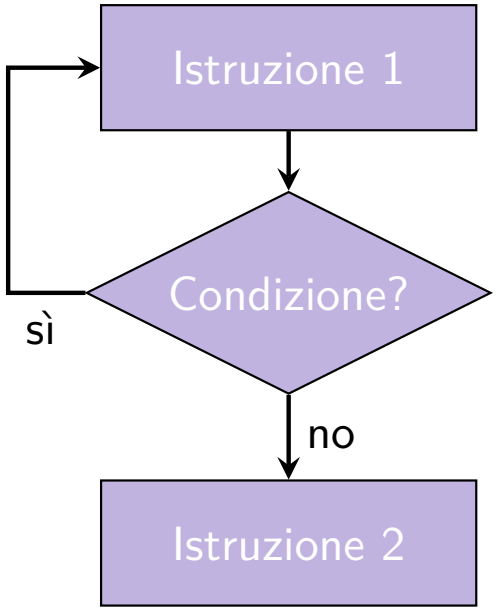


Iterazione

Pseudocodifica

```
esegui
   Istruzione 1
ripeti mentre condizione
```

Blocchi



Dai codici ai linguaggi

La scrittura binaria è molto comoda e semplice da gestire per una macchina (0 = circuito chiuso, 1 = circuito aperto).

Un programma (ovvero un insieme di algoritmi) per poter essere eseguito da una macchina deve essere scritto in linguaggio binario.

Intuiamo tuttavia che scrivere un programma in codice binario è molto complesso per un essere umano, ed è per questo motivo che sono stati inventati i linguaggi di programmazione.

I linguaggi di programmazione permettono di scrivere algoritmi con un linguaggio più "vicino" a quello che parliamo.

Dai codici ai linguaggi

La scrittura binaria è molto comoda e semplice da gestire per una macchina (0 = circuito chiuso, 1 = circuito aperto).

Un programma (ovvero un insieme di algoritmi) per poter essere eseguito da una macchina deve essere scritto in linguaggio binario.

Intuiamo tuttavia che scrivere un programma in codice binario è molto complesso per un essere umano, ed è per questo motivo che sono stati inventati i linguaggi di programmazione.

I linguaggi di programmazione permettono di scrivere algoritmi con un linguaggio più "vicino" a quello che parliamo.

Dai codici ai linguaggi

La scrittura binaria è molto comoda e semplice da gestire per una macchina (0 = circuito chiuso, 1 = circuito aperto).

Un programma (ovvero un insieme di algoritmi) per poter essere eseguito da una macchina deve essere scritto in linguaggio binario.

Intuiamo tuttavia che scrivere un programma in codice binario è molto complesso per un essere umano, ed è per questo motivo che sono stati inventati i linguaggi di programmazione.

I linguaggi di programmazione permettono di scrivere algoritmi con un linguaggio più "vicino" a quello che parliamo.

Dai codici ai linguaggi

La scrittura binaria è molto comoda e semplice da gestire per una macchina (0 = circuito chiuso, 1 = circuito aperto).

Un programma (ovvero un insieme di algoritmi) per poter essere eseguito da una macchina deve essere scritto in linguaggio binario.

Intuiamo tuttavia che scrivere un programma in codice binario è molto complesso per un essere umano, ed è per questo motivo che sono stati inventati i linguaggi di programmazione.

I linguaggi di programmazione permettono di scrivere algoritmi con un linguaggio più "vicino" a quello che parliamo.

Linguaggi di programmazione

I LdP sono particolari linguaggi artificiali che vengono utilizzati nella comunicazione umano-computer.

Le caratteristiche di un linguaggio di programmazione sono:

- un vocabolario ristretto (si utilizzano poche parole semplici);
- regole di costruzione delle istruzioni molto semplici e rigide;
- l'utilizzo di strutture predeterminate (come quelle viste).

Esempi di linguaggi di programmazione tra i circa 2500 esistenti: Fortran (1957), Pascal (1970), C++ (1986), Python (1991), JavaScript (1995, usato nel 98% dei siti web).

Linguaggi di programmazione

I LdP sono particolari linguaggi artificiali che vengono utilizzati nella comunicazione umano-computer.

Le caratteristiche di un linguaggio di programmazione sono:

- un vocabolario ristretto (si utilizzano poche parole semplici);
- regole di costruzione delle istruzioni molto semplici e rigide;
- l'utilizzo di strutture predeterminate (come quelle viste).

Esempi di linguaggi di programmazione tra i circa 2500 esistenti: Fortran (1957), Pascal (1970), C++ (1986), Python (1991), JavaScript (1995, usato nel 98% dei siti web).

Linguaggi di programmazione

I LdP sono particolari linguaggi artificiali che vengono utilizzati nella comunicazione umano-computer.

Le caratteristiche di un linguaggio di programmazione sono:

- un vocabolario ristretto (si utilizzano poche parole semplici);
- regole di costruzione delle istruzioni molto semplici e rigide;
- l'utilizzo di strutture predeterminate (come quelle viste).

Esempi di linguaggi di programmazione tra i circa 2500 esistenti: Fortran (1957), Pascal (1970), C++ (1986), Python (1991), JavaScript (1995, usato nel 98% dei siti web).

Linguaggi di programmazione

I LdP sono particolari linguaggi artificiali che vengono utilizzati nella comunicazione umano-computer.

Le caratteristiche di un linguaggio di programmazione sono:

- un vocabolario ristretto (si utilizzano poche parole semplici);
- regole di costruzione delle istruzioni molto semplici e rigide;
- l'utilizzo di strutture predeterminate (come quelle viste).

Esempi di linguaggi di programmazione tra i circa 2500 esistenti: Fortran (1957), Pascal (1970), C++ (1986), Python (1991), JavaScript (1995, usato nel 98% dei siti web).

Linguaggi di programmazione

I LdP sono particolari linguaggi artificiali che vengono utilizzati nella comunicazione umano-computer.

Le caratteristiche di un linguaggio di programmazione sono:

- un vocabolario ristretto (si utilizzano poche parole semplici);
- regole di costruzione delle istruzioni molto semplici e rigide;
- l'utilizzo di strutture predeterminate (come quelle viste).

Esempi di linguaggi di programmazione tra i circa 2500 esistenti: Fortran (1957), Pascal (1970), C++ (1986), Python (1991), JavaScript (1995, usato nel 98% dei siti web).

Linguaggi di basso livello

Il linguaggio macchina è quello direttamente compreso e utilizzato dalla CPU ed è formato solo da 0 e 1.

Un linguaggio di basso livello è più semplice del linguaggio macchina, ma è comunque molto lontano dai linguaggi che usiamo oggi per programmare, perché è difficile da comprendere per un umano.

Un esempio di linguaggio di basso livello è assembly, che usa istruzioni come:

```
05 id ADD EAX, imm32
```

Linguaggi di basso livello

Il linguaggio macchina è quello direttamente compreso e utilizzato dalla CPU ed è formato solo da 0 e 1.

Un linguaggio di basso livello è più semplice del linguaggio macchina, ma è comunque molto lontano dai linguaggi che usiamo oggi per programmare, perché è difficile da comprendere per un umano.

Un esempio di linguaggio di basso livello è assembly, che usa istruzioni come:

```
05 id ADD EAX, imm32
```

Linguaggi di basso livello

Il linguaggio macchina è quello direttamente compreso e utilizzato dalla CPU ed è formato solo da 0 e 1.

Un linguaggio di basso livello è più semplice del linguaggio macchina, ma è comunque molto lontano dai linguaggi che usiamo oggi per programmare, perché è difficile da comprendere per un umano.

Un esempio di linguaggio di basso livello è assembly, che usa istruzioni come:

```
05 id ADD EAX, imm32
```

Linguaggi di alto livello

I linguaggi di alto livello utilizzano un linguaggio pseudo-umano, che rende più facile la scrittura e la verifica del corretto funzionamento.

I linguaggi di programmazione di alto livello utilizzano come base la lingua inglese.

Esempio di istruzione in C++:

```
int num1, num2, differenza;
cout « "Due numeri:  ";
cin » num1 » num2;
differenza = num1 - num2;
cout « "Risultato = " « differenza « endl;
```

Linguaggi di alto livello

I linguaggi di alto livello utilizzano un linguaggio pseudo-umano, che rende più facile la scrittura e la verifica del corretto funzionamento.

I linguaggi di programmazione di alto livello utilizzano come base la lingua inglese.

Esempio di istruzione in C++:

```
int num1, num2, differenza;
cout « "Due numeri:  ";
cin » num1 » num2;
differenza = num1 - num2;
cout « "Risultato = " « differenza « endl;
```

Linguaggi di alto livello

I linguaggi di alto livello utilizzano un linguaggio pseudo-umano, che rende più facile la scrittura e la verifica del corretto funzionamento.

I linguaggi di programmazione di alto livello utilizzano come base la lingua inglese.

Esempio di istruzione in C++:

```
int num1, num2, differenza;
cout « "Due numeri:  ";
cin » num1 » num2;
differenza = num1 - num2;
cout « "Risultato = " « differenza « endl;
```

Traduzione in linguaggio macchina

La macchina non può eseguire direttamente le istruzioni scritte in un linguaggio di alto livello.

È dunque necessario un "interprete" che traduca il programma sorgente (in linguaggio di alto livello) in istruzioni di macchina.

La traduzione contemporanea all'esecuzione del sorgente è spesso piuttosto lenta, e pertanto si utilizzano dei compilatori.

Traduzione in linguaggio macchina

La macchina non può eseguire direttamente le istruzioni scritte in un linguaggio di alto livello.

È dunque necessario un "interprete" che traduca il programma sorgente (in linguaggio di alto livello) in istruzioni di macchina.

La traduzione contemporanea all'esecuzione del sorgente è spesso piuttosto lenta, e pertanto si utilizzano dei compilatori.

Traduzione in linguaggio macchina

La macchina non può eseguire direttamente le istruzioni scritte in un linguaggio di alto livello.

È dunque necessario un "interprete" che traduca il programma sorgente (in linguaggio di alto livello) in istruzioni di macchina.

La traduzione contemporanea all'esecuzione del sorgente è spesso piuttosto lenta, e pertanto si utilizzano dei compilatori.

Compilatori e interpreti

Il compilatore è un programma (scritto in linguaggio macchina) in grado di leggere le istruzioni del sorgente, verificarne la correttezza linguistica e sviluppare automaticamente le corrispondenti istruzioni in codice macchina.

Il codice ottenuto, che la macchina può eseguire direttamente, è detto eseguibile o programma oggetto.

Non tutti i linguaggi richiedono la compilazione: alcuni, come JavaScript, possono essere eseguiti con traduzione simultanea. Sono detti linguaggi interpretati.

Compilatori e interpreti

Il compilatore è un programma (scritto in linguaggio macchina) in grado di leggere le istruzioni del sorgente, verificarne la correttezza linguistica e sviluppare automaticamente le corrispondenti istruzioni in codice macchina.

Il codice ottenuto, che la macchina può eseguire direttamente, è detto eseguibile o programma oggetto.

Non tutti i linguaggi richiedono la compilazione: alcuni, come JavaScript, possono essere eseguiti con traduzione simultanea. Sono detti linguaggi interpretati.

Compilatori e interpreti

Il compilatore è un programma (scritto in linguaggio macchina) in grado di leggere le istruzioni del sorgente, verificarne la correttezza linguistica e sviluppare automaticamente le corrispondenti istruzioni in codice macchina.

Il codice ottenuto, che la macchina può eseguire direttamente, è detto eseguibile o programma oggetto.

Non tutti i linguaggi richiedono la compilazione: alcuni, come JavaScript, possono essere eseguiti con traduzione simultanea. Sono detti linguaggi interpretati.