FONDAMENTI DI PROGRAMMAZIONE

Come rappresentare i numeri naturali? 1
-Algoritmo del modulo e della divisione
Come rappresentare i numeri interi? 2
-Modulo e segno
-Complemento a due
-Bias (o polarizzazione)
Come rappresentare i numeri reali? 3
-Virgola fissa
-Virgola mobile
Definizioni di base 5
Tipi fondamentali 7
Istruzioni strutturate 9
-Istruzioni condizionali
-Switch
-Istruzioni ripetitive
Stream: ingresso e uscita 10
Leggere dati da file 11
Funzioni 12
-Gestione a stack della memoria
-Funzione ricorsiva
Tipi derivati 13
Riferimenti 13
-Funzioni con il riferimento
Puntatori 15
-Funzioni con i puntatori come argomenti formali
-Puntatori come ritorno di funzioni
Array (vettore) 16
-Array e puntatori
-Array multidimensionali (matrici)
-Funzioni con array
-Matrice a funzione
Stringhe 18
-Funzioni con le stringhe
Strutture 19
-Strutture con strutture
Memoria dinamica 21
-Vettori con dimesione nota a tempo di esecuzione
-Matrici nella memoria dinamica (2 metodi)
Liste 23
-Stampare, inserire e estrarre con le liste
Pila 24
-Operazioni con la pila
Coda 25
-Operazioni con la coda
Ordinare un array 27
-Ordinamento per selezione (selectionSort)
-Ordinamento a bolla (bubbleSort)
-Array di stringhe
-Ricerca di un elemento in un vettore
Visibilità 28

Unità di compilazione 29
Operatore unario (risoluzione di visibilità) 29
Spazio dei nomi 30
Collegamento 30
Classi di memorizzazione 31
Programmazione a moduli **31**
Operazioni sulle liste **32**
-Eliminare un valore a scelta dalla lista
-Liste ordinate
Argomenti default delle funzioni 33
Classi **34**
-Ritorno del tipo classe
-Assegnamento membro a membro
Puntatore this 36
Visibilità a livello di classe 37
Costruttori **37**
-Costruttori per allocare memoria dinamica
-Costruttori di oggetti dinamici
-Costruttori di copia
Distruttori **39**
Regole di chiamata di costruttori e distruttori 39
Unioni **40**
Puntatori a funzione 41
Memoria dinamica finita? Set new handler 41
Funzioni friend 42
Overloading di operatori **42**
-Operatore incremento
-Simmetria tra operatori
-Operatore di assegnamento
-Operatori che si possono ridefinire
Espressioni letterali 44
Classi per l'ingresso e l'uscita **44**
-Manipolatori
-Controllo sugli stream
Liste di inizializzazione **46**
-Costanti e riferimenti all'interno delle classi
-Membro classe dentro ad una classe
Array di oggetti classe 47
Conversione **47**
-Costruttori di conversione
-Operatori di conversione
Membri statici 48
Funzioni const 49
Preprocessore **50**
-Compilazione condizionale
Ricerca lineare 51
Ricerca binaria 51
Architettura di Von Neumann (1946) 53
Assembler **54**
Linux **59**
-Comandi (terminale)

PC → Hardware: • case
• processore CPU
• memoria: → ROM (BIOS, carica il SO), solo lettura
→ RAM (SO e programmi), veloce e temporanea
→ Hard disks
→ penne USB

Nel calcolatore è possibile ricreare ogni tipo di informazione attraverso una SEQUENZA BINARIA (bit 0 e 1). Esso è l'unità di misura dell'informazione.

BYTE = 8 bit

Ogni sequenza indica un'informazione differente
Esempio: 01000001 può essere → 65
→ A maiuscola
→ è colore di un pixel

CODICE ASCII → è una codifica standard su 7 bit (il primo bit (CODIFICA) del byte è sempre 0)

COME RAPPRESENTARE I NUMERI NATURALI?

NOTAZIONE POSIZIONALE: $(25)_{dieci} = 2 \cdot 10^1 + 5 \cdot 10^0$
$(11001)_{due} = 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = (25)_{dieci}$

FORMULA DELLA SOMMATORIA:

$$N = \sum_{p_0}^{p-1} a_i \beta^i = a_{p-1} \beta^{p-1} + a_{p-2} \beta^{p-2} + \dots a_2 \beta^2 + a_1 \beta + a_0$$

a_{p-1}: CIFRA PIÙ SIGNIFICATIVA
a_0: CIFRA MENO SIGNIFICATIVA

ALGORITMO DEL MODULO E DELLA DIVISIONE:

mod = resto
div = quoziente divisione

$q_0 = N$
$q_1 = q_0 \, div \, \beta \quad a_0 = q_0 \, mod \, \beta$
$q_2 = q_1 \, div \, \beta \quad a_1 = q_1 \, mod \, \beta$

$q_p = q_{p-1} \, div \, \beta \quad a_{p-1} = q_{p-1} \, mod \, \beta$
fino a $q_p = 0$

INTERVALLO DI RAPPRESENTABILITÀ: $\beta^p - 1$

a_i = valore cifra
p = numero cifre
β = base

Per passare, ad esempio, dalla base 5 alla base 9, è necessario usare prima la SOMMATORIA per arrivare alla base 10, e poi il metodo mod/div per arrivare alla base 9.

BASI IN POTENZE DI DUE: (657)otto (A03)sedici

(110 101 111)due (1010 0000 0011)due

- Basta "trasformare" ogni cifra nel proprio corrispondente dell'altra base.

OVERFLOW: problema che interviene quando una cifra che si vuole rappresentare esce fuori dall'intervallo di rappresentabilità.

COME RAPPRESENTARE I NUMERI INTERI?

MODULO E SEGNO: Trasformare a in A:

A = (segno_a, ABS_a)

↳ es, ABS di 3 = 11

0 se è positivo

1 se è negativo

Esempio: rappresentare +3 su 6 bit.

A = segno_+3, ABS_+3 = 0011

segno → ABS

- Il problema di questa rappresentazione è che si spreca una combinazione, poiché +0 = -0.

Intervallo in modulo e segno: $[-(2^{p-1}-1), +2^{p-1}-1]$

↳ (stessa regola dell'intervallo normale nei numeri solo naturali. Però p-1, perché un bit è per il segno).

COMPLEMENTO A DUE: A = (a ≥ 0)? ABS_a : (due^p - ABS_a)

- Se il numero da rappresentare è maggiore di zero, se ne fa la rappresentazione come un normale numero naturale
- Se invece è minore di zero, ci si sottrae ABS diventa (due^p - ABS_a).
- In questo modo si risolve il problema del doppio zero.
- Per fare l'inverso si controlla la prima cifra. Se è 0, ti passa diretto ad a se è 1, a = -(due^p - A).

a = (a_{p-1} == 0)? +A : -(due^p - A)

Intervallo in complemento a due: $[-2^{p-1}, +2^{p-1}-1]$

BIAS (O POLARIZZAZIONE): $A = a + bias$, $bias = (2^{p-1} - 1)$

- In sostanza, in questa rappresentazione è come se il numero più piccolo dell'intervallo diventasse lo zero.

Esempio: $a = 0 \Rightarrow 2^{q-1} - 1 = 7$ $A =$ [illegible]0111

- per ottenere a inverso, è necessario fare $a = A - bias$, ossia $a = A - (2^{p-1} - 1)$

Intervallo in bias: $[-2^{p-1} - 1, 2^{p-1}]$, ossia $[-bias, bias + 1]$

- Il vantaggio del complemento a due, il più usato per i numeri interi, è che viene usata la stessa somma dei numeri naturali, senza dover ricorrere ad una macro circuiteria.

- Si parla di OVERFLOW quando sommando due numeri dello STESSO SEGNO, si ottiene un risultato di SEGNO OPPOSTO!

COME RAPPRESENTARE I NUMERI REALI?

- I bit possibili ci indicano la DENSITÀ e il range di cifre che si possono scrivere.

VIRGOLA FISSA:
- C'è un numero fisso di bit per la parte intera e un numero fisso per la parte frazionaria.
- $F(r)$ = parte frazionaria di r
  $I(r)$ = parte intera di r
- Si hanno
  - p bit per rappresentare r
  - f bit per rappresentare $F(r)$
  - $p - f$ bit per rappresentare $I(r)$
- la parte intera $I(r)$ si rappresenta con le tecniche note
- la parte frazionaria ha bisogno di un algoritmo

$f_0 = F(r)$

Se $f_0 \neq 0$:

$f_{-1} = F(f_0 * 2)$ $a_{-1} = I(f_0 * 2)$

finché $f_j = 0$ o si è raggiunta la precisione desiderata

ERRORE di TRONCAMENTO In alcuni casi si possono ottenere dei numeri periodici, senza che f_j sia mai uguale a zero. In questi casi, il risultato avrà un margine di errore rispetto a r.