Esame orale

Fondamenti di Informatica I

DISPENSA

**Modulo speciale: RECORD DI ATTIVAZIONE**

Il record di attivazione corrisponde ad una “memory zone”, contenente le informazioni necessarie all’esecuzione di una funzione.

L’invocazione di una funzione, infatti, determina:

la **creazione** **dell’istanza** (richiesta creata sotto forma di funzione in questo caso),

l’allocazione (cioè **l’assegnazione**) dello spazio di memoria per **parametri** e **variabili** **locali**

il **passaggio** **dei parametri** (assegnando i parametri locali a quelli generici della funzione)

l’**esecuzione del codice**

COMPOSIZIONE

Il record di attivazione non fa altro che localizzare nella memoria tali informazioni, compresi i **parametri,** le **variabili,** l’**indirizzo di ritorno** e il **collegamento dinamico.**

L’indirizzo di ritorno o **RETURN ADDRESS – RA** indica il punto della funzione chiamante da cui ripartire una volta terminato il ruolo della funzione chiamata.

Il collegamento dinamico o **DYNAMIC LINK** indica il punto finale del record di attivazione corrente, cioè di quella data funzione.

LA DIMENSIONE del record di attivazione varia di funzione in funzione, ma per le singole funzioni rimane costante e calcolabile a priori.

IL SUO CICLO DI VITA ha inizio con l’allocazione della funzione in memoria, prosegue per tutto il tempo di esecuzione e viene deallocato, cioè distrutto, al termine.

Ogni chiamata di funzione genera nuovi record anche per funzioni identiche, risultando perciò specificamente generati per quella data chiamata.

Funzioni chiamanti funzioni generano sequenze di record di attivazione; la specifica area di esecuzione viene perciò trattata come una pila, o per meglio dire una STACK (“stock”, generalmente secondo il metodo LIFO – LAST IN FIRST OUT. Tale metodo sottolinea il fatto che al momento della de-allocazione, l’ordine seguito è quello inverso rispetto all’allocazione, per cui l’ultimo record di attivazione allocato risulta essere il primo deallocato.

LE SEZIONI DELLA MEMORIA sono distinguibili in quattro differenti parti legate all’esecuzione di un programma:

il primo è proprio lo **stack**, la pila contenente i record di attivazione

il **code segment** contiene il codice da eseguire

il **data segment**, come dice il termine, contiene i dati globali a tutte le funzioni

l’**heap** contiene i dati dinamici generati in corso di svolgimento della funzione

Quando viene restituito un risultato, tale valore viene memorizzato utilizzando un registro della CPU (central processing unit – unità d’esecuzione) o inserendo uno spazio aggiuntivo nel record di attivazione stesso.

**Modulo speciale: LA CODIFICA**

Ogni esecutore automatico deve essere capace di memorizzare e manipolare dati ed istruzioni. Queste ultime, infatti, per essere adoperate, devono essere codificate, cioè rese leggibili dall’esecutore.

I metodi di codifica sono vari e differenti, e variano sia per l’alfabeto adoperato che per i metodi di combinazione usati, detti anche stati.

Ogni configurazione ha associata una particolare entità di informazione.

Come qualsiasi altro linguaggio, anche il linguaggio di codifica è composto e possiede alcune sezioni fondamentali:

l’alfabeto, cioè i simboli letterali o meno che si andranno ad utilizzare

la sintassi, ovvero le regole con cui si andranno a comporre i codici, permettendo la distinzione della simbologia corretta da quella errata

la semantica, ovvero il significato di una data informazione, e in particolare, l’entità di tale informazione

La variazione di una sola di queste componenti definisce un nuovo metodo di codifica.

LA CODIFICA BINARIA

Come dice il nome stesso, tale codifica adopera un alfabeto composto da due soli simboli, una caratteristica utile a diminuire la probabilità di errore.

Si tratta della codifica più usata in campo informatico, da cui deriva in un certo senso l’idea di BIT o BINARY DIGIT, l’unità di misura elementare dell’informazione rappresentabile con dispositivi elettronici, attraverso cui sono rappresentabili due stati. Aumentando i bit, ovviamente aumentano anche gli stati rappresentabili, pari a 2K per K bit; da qui possiamo anche dire che il logaritmo in base due di N stati è pari a K bit.

Con K bit possiamo rappresentare numeri da 0 a 2K-1.

LA CONVERSIONE DEI NUMERI INTERI da un sistema n-ario a quello binario avviene attraverso la divisione intera del codice in codifica n-aria per 2, la base della codifica binaria, fino al numero 1 (risultato 0, resto 1). I vari resti, riscritti dall’ultimo al primo, definiscono la corrispettiva forma di quel codice secondo la codifica binaria. Tale riscrittura corrisponde anche all’espressione di un dato numero mediante le potenze di 2.

In generale, quindi, la conversione avviene attraverso una serie di divisioni del numero scritto in una data base con il numero che rappresenta la base in cui si desidera convertirlo.

Per quanto riguarda la PARTE FRAZIONARIA, la conversione in base binaria avviene moltiplicando tale sezione (n) del numero per 2, sottraendovi 1 se n>1. Si continua in questo modo fino a che n risulti pari a 0 o ad un numero già visto in precedenza; a quel punto basterà riportare come codice “0, la parte intera ottenuta durante la sequenza di divisioni/sottrazioni”, indicando come periodici gli interi dei numeri compresi tra la prima occorrenza del numero decimale che si ripete (se c’è) e la prima riscontrata successivamente.

TIPI DI CODIFICHE

* MODULO E SEGNO
* COMPLEMENTO A 1
* COMPLEMENTO A 2

MODULO E SEGNO: SCHEDA ANALITICA

SEGNO: rappresentato esplicitamente (0 = + | 1= -)

CORPO: i bit successivi al primo indicano il valore assoluto del numero in codifica binaria

PRO: il bit del segno ha posizione libera

CONTRO: non sono applicabili le operazioni, e lo zero è rappresentabile in due modi differenti

COMPLEMENTO A 1: SCHEDA ANALITICA

SEGNO: i numeri negativi vengono rappresentati invertendo i bit dei loro corrispondenti positivi

CORPO: codifica binaria

PRO: non bisogna distinguere alcun bit legato solo al segno del numero

CONTRO: non sono applicabili le operazioni, e lo zero è rappresentabile in due modi differenti

COMPLEMENTO A 2:

Si tratta di una vera e propria fusione tra i due metodi precedenti:

SEGNO: il primo bit rappresenta automaticamente il segno del numero secondo le regole di MODULO E SEGNO (0 = + | 1= -)

CORPO: segue le regole del complemento a 1; per i numeri positivi corrisponde alla semplice applicazione della formula seguente, rispettando il quantitativo di bit disponibili

CK(x) = 2K + x

per i corrispondenti negativi si procede con l’inversione dei bit.

PRO: la codifica di 0 è univoca, e l’intervallo dei numeri rappresentabili con tale metodo comprende più possibilità; dati K bit, possiamo rappresentare [-2K-1, +2K-1-1]

CODIFICA DEI NUMERI RAZIONALI

* FIXED POINT – rappresentazione che prevede la conversione del numero in una coppia di interi, distinti in parte intera e parte razionale
* FLOATING POINT – rappresentazione anche detta “a virgola mobile”, che prevede anche in questo caso la conversione del numero in una coppia di interi; il primo numero è il numero stesso, reso intero, come se lo associassimo alla corrispettiva potenza del dieci; il secondo è l’esponente della potenza di dieci che gli abbiamo precedentemente associato.

Si parla di OVERFLOW o di UNDERFLOW quando il risultato ottenuto è prodotto attraverso un valore non rappresentabile, rispettivamente perché supera la quantità massima di bit disponibili alla rappresentazione e perché non raggiunge la quantità minima di bit utili alla rappresentazione

CODIFICHE DI CARATTERI: ASCII E UNICODE

Si tratta di due codifiche comuni, utili in particolare per la codifica dei caratteri, distinguibili per la quantità di bit adoperati:

l’ASCII adopera 7 bit per un totale di 125 caratteri, estesi a 256 (8 bit) in alcuni codici

l’UNICODE utilizza 16 bit, di cui i primi sette identici alla codifica ASCII, per un totale di 65536 caratteri totali, di cui 65408 non posseduti dalla codifica ASCII.

LA COMPRESSIONE DEI DATI consiste in un metodo di risparmio di risorse durante la memorizzazione e la trasmissione di dati. Tale conversione avviene sulla base di determinate caratteristiche proprie del codice; ad esempio, nel caso della compressione per frequenza, anzicchè assegnare ad ogni carattere alfanumerico un determinato numero di bit, l’assegnazione avviene in modo inversamente proporzionale alla frequenza con cui quei caratteri si ripetono. Ovviamente, le nuove sequenze devono essere codificabili, altrimenti il lavoro risulta totalmente inutile e insensato.

Il processo di codifica non è limitato a testi e numeri: viene attuato anche per immagini, suoni, filmati, istruzioni e dati di ogni genere.

Per quanto riguarda le immagini, in particolare, la loro codifica prevede dapprima la **discretizzazione**, ovvero la rappresentazione della medesima immagine in pixel (PICTURE ELEMENT), attraverso la creazione di una specie di griglia di righe e colonne quadrate sovrapposta all’immagine; ovviamente la qualità dell’immagine è direttamente proporzionale al numero di questi ultimi, che geometrizzando l’immagine, delineeranno meglio i vari tratti divenendo quantitativamente maggiori e dimensionalmente minori.

Nel caso di una scala di grigi, con K bit possiamo rappresentare 2K livelli di grigio. K sarà anche il numero di bit necessario per ogni pixel.

Sia i COLORI che la RISOLUZIONE (precisione e definizione dell’immagine) sono variabili necessarie a determinare la quantità di pixel necessari.

Ad ogni pixel è inoltre associato un codice, che a seconda delle codifiche adottate, permette di rappresentare i vari gradi della scala dei grigi, o, se definita, su una scala di colori differenti.

La codifica basata sulla rappresentazione di un’immagine in pixel è denominata BITMAP.

LE TECNICHE DI COMPRESSIONE LOSSY permettono di diminuire la grandezza di un’immagine approssimandone per difetto la quantità di byte prevista. Ovviamente la qualità ne risentirà, ma sarà quasi impercettibile per approssimazioni corrette. Tipi di compressione lossy sono ad esempio la conversione in formato JPEG o PNG.

I FILMATI sono convertiti come se si trattasse di sequenze di fotogrammi.

I SUONI sono campionati, cioè misurati ad intervalli nel caso di suoni analogici. Ovviamente, così come per i pixel, minore è l’intervallo di campionamento, maggiore è la qualità del suono.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **TIPI** | **TUPLE** | **DICTIONARY** | **STRING** | **SET** | **LIST** |
| **MUTABILITÀ** | IMMUTABILE | MUTABILE | IMMUTABILE | MUTABILE | MUTABILE |
| **DELINEATORI** | TONDE | GRAFFE | ‘-‘ o “-“ | GRAFFE | QUADRE |
| **INNESTAZIONI** | Sì | Sì |  |  | Sì |
| **TIPI ACCETTATI** | ETEROGENEI | ETEROGENEI | ETEROGENEI | PRIMITIVI | ETEROGENEI |
| **FUNZIONE DI CONVERSIONE** | tuple() | dict(elemento a coppie), per ottenere le componenti principali  nome\_del\_dict.keys()  nome\_del\_dict.values() | string() | set() | list() |
| **PRO** |  | Adopera l’hashing, veloce su inserimenti e rimozioni |  | Adopera l’hashing, veloce su inserimenti e rimozioni, elimina i duplicati |  |
| **CONTRO** |  | Non ha ordine, non compie confronti.  Le funzioni di max() e min() sono applicate sulle chiavi, come probabilmente altre funzioni built-in. |  |  |  |
| **PARTICOLARITÀ** |  | È costituito da coppie di chiavi (uniche) e valori, che velocizzano l’identificazione degli elementi |  | Potrebbero essere utilizzato come filtro dei duplicati |  |