15:56:58 From USER1: Ma definire una typedef su un intero non può causare confusione per chi usa quel tipo? Io so che operazioni posso fare su un intero, e che funzioni sono disponibili per manipolarlo, mentre un tipo età non so a priori come trattarlo, no?
16:04:25 From USER2 : 2?
16:04:34 From USER2 : intendo il pointer
16:14:57 From USER3: una sorta di pausa velata
16:15:21 From USER4: Ma come mai Pila *s ha assunto il valore 100?
16:15:34 From USER5 : l'ha messo lui a caso
16:15:45 From USER5 : tanto per mettere un valore
16:15:59 From USER4: E il 5 uguale?
16:16:03 From USER5 : sì
16:16:06 From USER6 : Ma 100 non era l'indirizzo?
16:16:07 From USER2 : Io invece vorrei sapere come si passa a s.marker, ossia se c'è un altro indirizzo di memoria oppure fa una differenza
16:16:16 From USER7 : Il 100 è un indirizzo
16:16:28 From USER8 : Si perche in teoria quando tu fai un new, dici al sistema operativo di riservarti uno spazio nella heap, e ti restituisce un puntatore
16:16:32 From USER9 : 100 e' scelto a caso ma ha valore di indirizzo
16:16:36 From USER8 : Per quello ha messo un valore arbitrario
16:16:39 From USER4: é l'indirizzo che punta s no?
16:16:41 From USER5 : esatto, un indirizzo a caso, intendevo
16:16:54 From USER5 : sì
16:17:05 From USER9 : eh si solo che magari dicendo che era solo a caso non si capiva bene
16:17:09 From USER4: Perfetto grazie mille
16:17:12 From USER10 : s.marker darebbe errore
16:17:17 From USER2 : La mia
16:17:36 From USER5 : sì, hai ragione: poteva essere ambiguo
16:17:52 From USER6 : Credo lo sappia in base alla struct
16:18:10 From USER10 : s è un puntatore, si deferenza tramite l'asterisco o la freccetta e si arriva a marker
16:18:16 From USER6 : Siccome si riserva tot memoria sa quanti byte avanti sta il campo che cerci o no?
16:18:37 From USER5 : per come è stata definita la struct Pila
16:24:20 From USER10 : Io non ho mai capito pienamente come fa delete[] a sapere quanti byte "distruggere", dato che non gli passo la grandezza del mio array.
16:24:39 From USER5 : no
16:24:41 From USER11 : lo sa il sistema, disse il prof ieri
16:25:28 From Michael Saccone : Viva il Sistema
16:25:31 From USER1: che però giustamente se il sistema lo sapesse allora perché devo tenermi salvata la grandezza dell'array salvata in una variabile, ancora non capisco
16:26:03 From USER5 : perché lo scopo di tutto questo è creare una pila a dimensione variabile.
16:26:04 From USER10 : Stessa domanda pure io.. sarà che è un'informazione non accessibile all'utente
16:26:05 From USER9 : probabilmente sono informazioni che non puoi accedere
16:26:05 From USER1: cioè, potrei fare come gli altri linguaggi, array.length(), o len(array), per menzionarne alcuni
16:26:16 From Riccardo Germenia : per sapere fino a che punto deve andare avanti il puntatore (se non sbaglio)

16:26:52 From USER11 : perché l'operazione di delete è one-time rispetto alle altre operazioni sulle array, che con un controllo sulla length "rovinerebbero" l'efficienza di C
16:27:05 From USER11 : (sto citando la spiegazione del prof di ieri a riguardo)
16:27:22 From USER1: ah, quindi la delete è l'unica che controlla la lunghezza?
16:27:36 From USER1: E allora perché non posso controllarla anche io quando mi serve?
16:28:02 From USER6 : perché C non lo permette
16:28:03 From USER11 : sempre per la spartanità di C che ti dice di vedertela te
16:28:47 From USER11 : tecnicamente con un buon programmatore è più efficiente in termini di prestazioni e sarebbe giustificabile
16:29:10 From USER1: ma non c'entra spartanità o no
16:29:20 From USER1: non è un controllo sulla lunghezza che mi serve
16:29:34 From USER1: è semplicemente ricavarla
16:30:18 From USER1: la spartanità di C è perché si può uscire dagli array, ma non spiega il perché non esiste una funzione array.length()
16:31:08 From USER6 : Bjarne Stroustrup non la voleva

In realtà si può fare questo:
https://stackoverflow.com/questions/37538/how-do-i-determine-the-size-of-my-array-in-c

16:31:27 From USER10 : boh secondo me quando c alloca dice al sistema quanto ha allocato e il sistema lo sa, e sa quanto liberare quando glielo si chiede. Ma non è permesso interrogare il sistema sulla lunghezza di quell'allocazione, almeno penso.
16:31:32 From USER1: a quanto pare la delete lo sa perché: "When you allocate memory on the heap, your allocator will keep track of how much memory you have allocated. This is usually stored in a "head" segment just before the memory that you get allocated. That way when it's time to free the memory, the de-allocator knows exactly how much memory to free."
16:33:10 From USER10 : boh pero capisco Ovidiu che tecnicamente se l'informazione è presente perché dovrei farmene una copia in una variabile anziché interrogarla?
16:34:54 From USER12: Perché non sai dove l'informazione è contenuta
16:35:18 From USER1: E allora non lo saprebbe nemmeno la delete
16:35:25 From USER11 : immagino che la delete iteri l'heap
16:35:29 From USER11 : fino alla head di cui parlava
16:35:50 From USER10 : Ahhh si potrebbe avere senso ciò che dice Deme
16:35:58 From USER1: Oddio, dubito che venga iterata la heap
16:36:14 From USER1: Ti rendi conto quanto sarebbe dispendioso liberare memoria
16:36:18 From USER10 : che quella head fosse una flag che dice a delete: "ok basta così hai eliminato abbastanza"?
16:36:24 From USER11 : mica tutta la heap
16:36:26 From USER11 : a partire dal puntatore
16:37:33 From USER12: Io penso che la delete dica al sistema di eliminare l'array, in seguito il sistema va a controllare la grandezza e infine dealloca l'array.
16:37:45 From USER12: quindi solo il sistema sa
16:37:47 From USER10 : tipo array in heap puntatore->[][][][][][][][][][stop flag]
16:38:01 From USER10 : il delete parte dal puntatore e arriva alla stop flag?

16:38:28 From USER8 : Qualcuno chiede in aula da parte mia?
16:38:30 USER13: Ma la chat vocale nemmeno all'ora prima funzionava se non sbaglio
16:38:31 From USER10 : l'unico modo è chiedere a quelli in aula
16:38:35 From USER1: Eh ma allora sarebbe come le C string, dove c'è un terminatore di stringa
16:38:38 USER14 : ma deallocare nell'heap libera quello spazio? intendo come nello stack si può riutilizzare quella memoria?
16:38:51 From USER11 : sì
16:38:59 From USER8 : Volevo chiedergli di realloc, del quale ci aveva parlato roveri se qualcuno si ricorda. Non si potrebbe usare quello al posto di quel casino di deallocazione e riallocazione?

Si ma noi vogliamo lavorare con new e delete, non con malloc & C. (poi capiremo il perché). Comunque sulla questione dell'allocazione dinamica memoria si veda questa risposta:

https://stackoverflow.com/questions/20363591/how-does-dynamic-memory-allocation-work/20363662#20363662

16:39:03 USER14 : perfect, thx
16:42:56 From USER8 : Qualcuno che chieda la mia domanda due messaggi sopra? XD
16:43:16 From USER11 : big F
16:43:27 From USER8 : Eh ma questo e' un complotto
16:43:34 USER15 : XD
16:43:34 From USER20: puoi riscriverla che non la vedo?
16:43:59 USER15 : Volevo chiedergli di realloc, del quale ci aveva parlato roveri se qualcuno si ricorda. Non si potrebbe usare quello al posto di quel casino di deallocazione e riallocazione?
16:44:24 From USER8 : Il prof Roveri ci aveva parlato dell'operatore "realloc", nativo del C e che il C++ ha ereditato in teoria. Non sarebbe meglio utilizzare quello piuttosto che new e delete?
16:45:52 From USER8 : Parlo dell'espansione della pila ovviamente
16:46:38 From USER1: F per Filippo
16:46:39 From USER8 : Sad story
16:50:05 From USER17: assert è una specie di if else?
16:50:16 From USER1: sembrerebbe di si
16:50:25 From USER1: solo che non capisco se restituisca qualcosa o sia proprio una exit() dalla funzione
16:50:42 From USER11 : c'è scritto
16:50:46 From USER11 : l'esecuzione termina
16:52:14 From USER10 : ah fa tipo manicare il programma
16:52:15 From USER1: ah proprio tutto il programma
16:52:18 From USER1: tipo die() su PHP
16:52:23 From USER10 : panicare*
16:52:45 From USER8 : Ma non e' l'equivalente di if (condizione) exit(1)?
16:53:00 From USER16 : si, sembra
16:53:00 From USER8 : 1 messo a caso per dire non andato a buon fine
16:53:00 From USER10 : si ma è più elegante
16:53:09 From USER8 : Ok
16:53:18 From USER11 : internamente probabilmente non è uguale comunque
16:53:21 From USER11 : visto che non specifichi un exit code
16:53:25 From USER18: Domanda: ma se termina il programma non c'è il rischio poi di un memory leak?

16:53:46 From USER1: no, perché appena il processo termina il sistema operativo libera comunque tutta la memoria

ESATTO

16:54:25 From USER1: Il problema esisterebbe se dopo che il programma ha terminato rimanesse un processo "zombie" che la tiene occupata, ma non è questo il caso
16:54:33 From USER17: questo liberare la memoria quando chiude il programma è una cosa esclusiva di assert=
16:54:34 From USER8 : Quando un programma termina il sistema operativo dealloca sempre tutto, quindi no worries. Roveri ci faceva deallocare tutto manualmente per buona pratica, perche poi quando per esempio creeremo un server web, meglio che ci ricordiamo di deallocare quello che non serve, se no bye memoria
16:54:34 From USER17: ?*
16:54:45 From USER17: ah ok
16:55:02 From USER18: ok thank's
16:55:14 From USER10 : viva il sistema
16:55:17 From USER1: No, non è una cosa di assert. Quando il programma termina, il sistema operativo libera tutto, sempre
16:55:29 From USER11 : sistema goes bonk
16:55:36 From USER9 : chiama 9 volte non 10
17:00:39 From USER1: bullshit
17:00:41 From USER1: that's what
17:01:00 From USER17: previsione del futuro azzeccata ovidiu ahahah
17:01:07 From USER8 : Ma che poi se hai fortuna trovi bullshit, se no trovi IL MARTELLO DEL BAN
17:03:27 From USER2 : shalow copy?
17:03:32 From USER2 : shallow*
17:03:45 From USER10 : forbidden copy
17:04:29 From USER2 : già
17:04:36 From USER16 : copiando l'indirizzo di memoria del primo elemento
17:04:37 From USER5 : sto creando un secondo puntatore alla stessa cosa?
17:04:41 From USER19: Copia l'indirizzo di memoria o?
17:04:46 From USER8 : Si
17:04:47 From USER11 : copia l'indirizzo
17:04:58 From USER2 : non è neanche shallow copy in realtà
17:05:06 From USER2 : chiedo venia
17:05:30 From USER10 : chi è Venia?
17:06:00 From USER1: Shallow copy: "In questo processo, B è allegato allo stesso blocco di memoria di A. Questo metodo è conosciuto anche come copia per indirizzo."
17:06:19 From USER10 : allora aveva ragione
17:06:45 From USER2 : Ho trovato una spiegazione in cui c'era scsritto che bisognava creare tte references a tutti gli elementi
17:07:23 From USER2 : delle references* quindi oltre che copiare il primo puntatore bisognava creare altri puntatori per il contenuto