```c
#include <stdio.h>
int main(){
  char iniz[8] = "-INIZIO";

  char*          pChar;
  unsigned char* pUC;
  short int*     pSI;

                              //--- un char occupa 1 byte
  char           si = 'S'; // 'S'=83 in codice ASCII dec.
  char           no = 'N'; //-- 'N'=78 in decimale
  char           x  =  1;  //---codice ASCII in decimale
  char           y  = 65;  //---codice ASCII di 'A'
  char           z  =  1;  //--carattere NON stampabile
  unsigned char  t  = 97;  //--carattere stampabile: 'a'

  short int SInt[4] = {1,2,3,4};     //--- 2*4 = 8 byte
  short int          xx = -32768;
  short int          yy = +32767; //1 short occupa 2B
  unsigned short int zz =  36,  // codice decimale di '$'
                     tt =  16192; //interpretabile come
  char stop[8] = "--FINE-"; // 2 char:  16192 = 16128 + 64
// cioè 63*2^8 +64*2^0 con 63 codifica in decimale di '?' e 64 di '@'
printf( "size of :\n");
printf( "--iniz   : %d\n" , sizeof(iniz) );
printf( "--pChar  : %d\n" , sizeof(pChar) );
printf( "--pUC    : %d\n" , sizeof(pUC)   );
printf( "--pSI    : %d\n" , sizeof(pSI)   );
printf( "--si     : %d\n" , sizeof(si)    );
printf( "--t      : %d\n" , sizeof(t)     );
printf( "--SI     : %d\n" , sizeof(SInt)  );
printf( "--tt     : %d\n" , sizeof(tt)  );

printf( "\nindirizzo di:\n" );
printf( "- iniz  : 0x%x = %d\n", &iniz , &iniz );
printf( "- pChar : 0x%x = %d\n" , &pChar, &pChar );
printf( "- pUC   : 0x%x = %d\n" , &pUC  , &pUC  );
printf( "- pSI   : 0x%x = %d\n" , &pSI  , &pSI  );
printf( "- si    : 0x%x = %d\n" , &si   , &si   );
printf( "- no    : 0x%x = %d\n" , &no   , &no   );
printf( "- x     : 0x%x = %d\n" , &x    , &x    );
printf( "- y     : 0x%x = %d\n" , &y    , &y    );
printf( "- z     : 0x%x = %d\n" , &z    , &z    );
printf( "- t     : 0x%x = %d\n" , &t    , &t    );
printf( "- SInt  : 0x%x = %d\n" , &Int  , &SInt );
printf( "- xx    : 0x%x = %d\n" , &xx   , &xx   );
printf( "- yy    : 0x%x = %d\n" , &yy   , &yy   );
printf( "- zz    : 0x%x = %d\n" , &zz   , &zz   );
printf( "- tt    : 0x%x = %d\n" , &tt   , &tt   );
printf( "- stop  : 0x%x = %d\n" , &stop , &stop );
printf( "\n" );
printf( "- iniz [2] : 0x%x = %d , val=%c\n"  ,
          &iniz[2] , &iniz[2] , iniz[2] );
printf( "- SInt [2] : 0x%x = %d , val=%hd\n" ,
                 &SInt[2] , &SInt[2] , SInt[2] );
printf( "- stop [2] : 0x%x = %d , val=%c\n" ,
                 &stop[2] , &stop[2] , stop[2] );
printf( "\n" );
```

Il programma (in linguaggio **C**) definisce 3 **puntatori** e **variabili** per i tipi *char, unsigned char e short int*; espone quindi l'**occupazione di memoria** per alcune variabili (array di *char* da 8 byte, singoli *char* da 1 byte, *short int* da 2 byte) e per i **puntatori** (sono **tutti da 4 byte**).

Espone quindi gli **indirizzi** delle varibili e dei puntatori in **esadecimale** (specificatore di formato **%x**) e poi in **decimale** (specificatore di formato **%d**).

```
Prompt dei comandi
>>
>>puntCharC
size of :
--iniz   : 8
--pChar  : 4
--pUC    : 4
--pSI    : 4
--si     : 1
--t      : 1
--SI     : 8
--tt     : 2

indirizzo di:
- iniz  : 0x60fefc = 6356732
- pChar : 0x60fef8 = 6356728
- pUC   : 0x60fef4 = 6356724
- pSI   : 0x60fef0 = 6356720
- si    : 0x60feef = 6356719
- no    : 0x60feee = 6356718
- x     : 0x60feed = 6356717
- y     : 0x60feec = 6356716
- z     : 0x60feeb = 6356715
- t     : 0x60feea = 6356714
- SInt  : 0x60fee2 = 6356706
- xx    : 0x60fee0 = 6356704
- yy    : 0x60fede = 6356702
- zz    : 0x60fedc = 6356700
- tt    : 0x60feda = 6356698
- stop  : 0x60fed2 = 6356690

- iniz [2] : 0x60fefe = 6356734 , val=N
- SInt [2] : 0x60fee6 = 6356710 , val=3
- stop [2] : 0x60fed4 = 6356692 , val=F
```

*NOTA: le variabili vengono allocate a partire dall'ultima variabile dichiarata: prima l'array **stop** all'indirizzo 6.356.690 di Memoria Centrale, dopo 8 byte la variabile **tt** (indirizzo 6.356.698) che occupa 2 byte, quindi la variabile **zz** (indirizzo 6.356.700), infine l'array **iniz**.*

Il programma espone poi **indirizzi** e **valori** degli elementi di posizione 2 degli array **iniz**, **SInt** e **stop**.

```
    pChar = &y;
    printf( "%c\n\n"   , *pChar );

    pChar = &x; //--carattere NON stampabile (SOH)
    printf( "%c\n\n"   , *pChar );

    pUC   = &t;
    printf( "%c\n\n"   , *pUC );

    pSI =   &xx;
    printf( "%d\n\n"   , *pSI );
//--- per short int specificatori di formato %d oppure %hd
    pSI =   &SInt[3];
    printf( "%hd\n\n"  , *pSI );
    printf( "\n" );

pChar = stop;       //--- indirizzo del 1^ elemento dell'array
for (int i=0; i<10; i++) //gli indirizzi aumentano di 1 in 1)
    printf( "%c.", *pChar++ );
printf( "\n%c.\n" , *pChar );
printf("- pChar : 0x%x = %d\n\n", pChar,pChar);
```

seguono **stop** *(***tt***); poi stampa il contenuto dell'ultimo indirizzo raggiunto da pChar e il valore dell'**indirizzo**, 6.356.700, corrispondente alla varibile* **zz**.

Scrive poi **6 valori** puntati dal puntatore **pSI** a *short int* partendo dall'array **SInt** *(in questo caso gli indirizzi aumentano di 2 in 2)*: vengono esposti i 4 elementi dell'array (1, 2, 3 e 4) e poi vengono interpretati come *short int* i successivi 4 byte corrispondenti alle coppie di *char* **t**,**z** e **y**,**x**. **t** (vale 97) è interpretato come byte meno significativo dello *short int* e **z** (=1) come byte più significativo (memorizzazione *little endian*), ovvero:

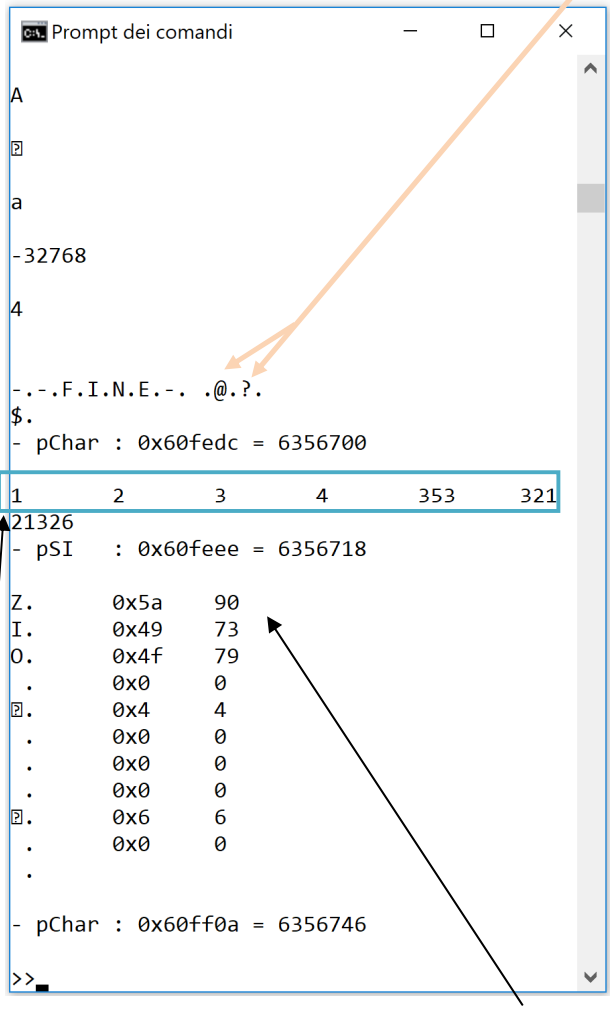
$$1*2^8 + 97*2^0 = 256 + 97 = 353$$

Analogamente per **y** e **x** e poi per **no** (= 78) e **si** (= 83):

$$83*2^8 + 78*2^0 = 21248 + 78 = 21326$$

```
pSI = SInt;
for (int i=0; i<6; i++)
    printf( "%hd\t" ,*pSI++ );
printf( "\n%hd\t" ,*pSI );
printf("\n- pSI   : 0x%x = %d\n\n" , pSI  , pSI);
pChar = &iniz[4];
for (int i=0; i<10; i++)
    printf( "%c.\t0x%x\t%d\n" ,
            *pChar++, *pChar , *pChar );
printf( "%c.\n\n" , *pChar );
printf( "- pChar : 0x%x = %d\n" , pChar, pChar );
}
```

Viene ora posto in **pChar** l'**indirizzo** della variabile **y** e se ne visualizza il **valore** (**contenuto** della **variabile puntata** da **pChar**); si procede in modo analogo per le variabili **x**, **t**, **xx** e per l'elemento di posto 3 dell'array **SInt**. Si visualizzano poi gli 8 elementi dell'array **stop**. *NOTA: il ciclo stampa **10** byte seguiti da un punto, quindi anche 2 byte che*



Infine vengono esposti 10 byte a partire da **iniz[4]** (='Z') in formato char (**%c**), e il codice ASCII in esadecimale (**%x**) e decimale (**%d**).

*NOTA: nelle istruzioni **printf** è presente l'istruzione di incremento unitario (**++**) che viene eseguito **solo dopo la fine della istruzione** di stampa del contenuto **puntato** dal **puntatore** (es: **pChar**). Dopo il carattere 'O' (**iniz[7]**) viene stampato il terminatore di stringa ('\0' : 0x0) e altri 7 byte successivi all'array* **iniz** *(ultimo indirizzo raggiunto, 6.356.746 > 6.356.732+8)*