dynamiczny przydział pamięci

malloc() free() realloc()

calloc() memset() memcpy( )

mempcpy( ) memmove()

(**wskaźniki!!**)

dynamiczny przydział pamięci

void * **memccpy** (*void * to, void * from, int c, int size*)

funkcja kopiuje nie więcej niż size bajtów z miejsca wskazywanego przez from do miejsca wskazywanego przez to, zatrzyma się jeśli napotka bajt zgodny z c; zwraca wskaźnik do miejsca docelowego jeden bajt za miejscem gdzie c zostało skopiowane;

jeśli błąd, to zwraca wskaźnik NULL

dynamiczny przydział pamięci

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void * wsk;

char a[7]={'0','1','2','3','4','5','6'};
char b[7]={'a','b','c','d','e','f','g'};
   int main()
{
   int n;
   printf("\n  a[2]=%1c  b[2]=%1c\n", a[2], b[2] );
   wsk = (void *) memccpy(b, a, (int)'3', 10);
```

dynamiczny przydział pamięci

```
for(n=0;n<=10;++n)
{ printf("  %c",b[n]);
}
printf("\n   *wsk=%c\n", *((char*)wsk) );
exit(0);
}/* koniec main */
```

dynamiczny przydział pamięci

a[2]=2 b[2]=c

0 1 2 3 e f g

*wsk=e

pointerb.c

Przypomnienie: to, że wskaźnik wskazuje na jakiś adres **nie gwarantuje tego,** że pod tym adresem przechowywane jest coś sensownego z punktu widzenia wykonującego się programu.

Przykładowy program: np. pointerb.c ,
http://users.uj.edu.pl/~ufrudy

(wystarczy np., że wskaźnik został załadowany adresem zmiennej, której czas życia minął, gdyż była zmienna klasy auto w jakiejś funkcji)

ile pamięci zaalokował malloc? jak uzyskać informację?

```
#include <malloc.h>

 struct mallinfo alex;

alex=mallinfo ( );   /* wywołanie funkcji  */

  /*   alex.arena          alex.hblkhd
całkowita ilość pamięci w bajtach alokowana przez malloc
przez sbrk oraz przez mmap  */
```

struct mallinfo

www.gnu.org/software/libc/manual/html_node/Statistics-of-Malloc.html

int arena

This is the total size of memory allocated with sbrk by malloc, in bytes.

int ordblks

This is the number of chunks not in use. (The memory allocator internally gets chunks of memory from the operating system, and then carves them up to satisfy individual malloc requests)

int smblks

This field is unused.

int hblks

This is the total number of chunks allocated with mmap.

struct mallinfo

www.gnu.org/software/libc/manual/html_node/Statistics-of-Malloc.html

hblks

This is the total number of chunks allocated with mmap.

int hblkhd

This is the total size of memory allocated with mmap, in bytes.

int usmblks

This field is unused.

int fsmblks

This field is unused.

struct mallinfo

www.gnu.org/software/libc/manual/html_node/Statistics-of-Malloc.html

int uordblks

This is the total size of memory occupied by chunks handed out by malloc.

int fordblks

This is the total size of memory occupied by free (not in use) chunks.

int keepcost

This is the size of the top-most releasable chunk that normally borders the end of the heap (i.e. the high end of the virtual address space's data segment).

struct mallinfo, mallinfo()

przykład użycia

```
#include <stdio.h>    /*   przykład mallinfo ()   */

#include <stdlib.h>


#include <malloc.h>
```

struct mallinfo, mallinfo()
przykład użycia

```
#include <stdio.h>    /*   przykład mallinfo ()   */
#include <stdlib.h>


#include <malloc.h>
```

struct mallinfo, mallinfo()
przykład użycia

```
struct st
 {
  float buf[1000*1000*10];
 }  *p1;


  struct mallinfo alex;
          main()
         {/*  w tej funkcji main jest użyta funkcja mallinfo */
```

struct mallinfo, mallinfo()
przykład użycia

```
 while(1)
  {   p1=malloc( sizeof(struct st) ); if(p1==NULL) { printf("..brak
pamieci..\n");      break;  }      alex=mallinfo();
    printf("\n   main      alex.arena=%d  alex.ordblks=%d  alex.hblks=%d",
                        alex.arena, alex.ordblks, alex.hblks);
    printf("\n              alex.hblkhd=%u  alex.uordblks=%d",
                        alex.hblkhd, alex.uordblks);
    printf("\n              alex.fordblks=%d  alex.keepcost=%d\n",
                        alex.fordblks, alex.keepcost);
    printf("\n");
  }                }      /*   koniec   funkcji   main  */
```

struct mallinfo, mallinfo()

przykład użycia

main alex.arena=0 alex.ordblks=1 alex.hblks=1

alex.hblkhd=40001536 alex.uordblks=0

alex.fordblks=0 alex.keepcost=0

.....

main po alex.arena=0 alex.ordblks=1 alex.hblks=12

alex.hblkhd=480018432 alex.uordblks=0

alex.fordblks=0 alex.keepcost=0

..... *(aż w końcu program wypisze)*

..brak pamieci..

argumenty wiersza poleceń

funkcja main ma dwa argumenty

```
main(int argc, char * argv [ ] )
```

(tak sa zwyczajowo nazywane, choć nie są to nazwy obowiązujące)

argumenty wiersza poleceń

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
 int   main  (int argc,  char *argv[] )
   /*    albo  int main( int argc,   char  **argv)    */
{ int n;
  for(n=0;n<argc; ++n)
  {  printf("\n            n=%d  param= %s",n, argv[n]);
     printf("\n to samo... n=%d  param= %s",n, *(argv+n) );
     printf("\n");
  }     exit(0);
 }   /*    koniec     funkcji     main     */
```

argumenty wiersza poleceń: getopt

```
#include <stdio.h>    /*   getopt2.c  */
#include <stdlib.h>
#include <unistd.h> /* niezbędne */
  int main(int argc, char *argv[])
{
     int n;
     int opt;
```

argumenty wiersza poleceń: getopt

```
printf("\n argc=%d\n",argc);
for(n=0;n<argc;++n)
{
 printf(" *argv[%d] %s\n",n, argv[n]);
}
printf("\n");
```

argumenty wiersza poleceń: getopt

```
while( (opt=getopt(argc,argv,"a:b:")) != -1 )
      { switch (opt)
        {
         case 'a' :
         printf("case \'a\'\n");
         printf("   optarg=%s=\n",optarg);
         break;
         case 'b' :
         printf("case \'b\'\n");
         printf("   optarg=%s=\n",optarg);
         break;
             }
      }  /*  koniec while  */
} /* koniec main */
```

argumenty wiersza poleceń: getopt

przykład: a.out -a45 -b789

argc=3

*argv[0] a.out

*argv[1] -a45

*argv[2] -b789

case 'a'

optarg=45=

case 'b'

optarg=789=