**Lab\_Memory\_Manager**

------------------------------------------------------------------------------------------------------------------------------------

Stworzyć manager pamięci, który składa się z:

1. Tablicy struktur MY\_MEMORY\_MANAGE \* MemTab, gdzie

struct MY\_MEMORY\_MANAGE

{

 void \* ptr; //wskaznik typu void \*

 size\_t numAlloc; //kolejny numer alokowania

};

1. size\_t TotalOccupedMem = 0; //Ogólnie alokowana pamięć, B.

size\_t last = 0; //pierwsza wolna pozycja w MemTab.

size\_t AllocCount = 0; //numer kolejnej alokacji.

Każdy element tablicy przechowuje adres, który zwracają funkcji malloc, realloc, a również numer wywołania funkcji alokowania/realokowania pamięci.

Manager pamięci obsługują funkcji:

int MyMemInit();

void MyMemClose();

void \* MyMemAlloc(size\_t NoBytes);

void \* MyMemRealloc(void \*ptr, size\_t NoBytes);

void MyMemFree(void \*ptr);

size\_t MyGetOccupedMem();

MyMemInit alokuje pamięć po raz pierwszy i powinna być wywołana na samym początku programu. Przyjmujemy, że pierwsze alokowanie tworzy tablicę o 5 elementów.

MyMemClose sprawdza, czy tablica MemTab jest pusta. Każdy element tablicy, zawierający ptr != NULL, świadczy o to, że pamięć była alokowana, jednak nie zwolniona. Wyprowadzamy komunikat na monitor i zwalniamy pamięć. Po zwolnieniu ptr dla wszystkich elementów tablicy zwalniamy samą tablicę MemTab. Sprawdzamy – TotalOccupedMem powinien być równym do zera.

MyMemAlloc – alokuje pamięć o NoBytes bajtów i umieszcza w pierwszy wolny element tablicy MemTab adres zwracany poprzez malloc oraz bieżący numer alokowania. Na przykład:

void \* MyMemAlloc(size\_t NoBytes)

{

 size\_t noitems = \_msize(MemTab) / sizeof(MY\_MEMORY\_MANAGE);

 if (last >= noitems)

 {

 MemTab = (MY\_MEMORY\_MANAGE \*)realloc(MemTab, 2 \* noitems \* sizeof(MY\_MEMORY\_MANAGE));

 if (MemTab)

 memset(MemTab + noitems, 0, noitems \* sizeof(MY\_MEMORY\_MANAGE));

 }

 MemTab[last].ptr = malloc(NoBytes);

 if (MemTab[last].ptr)

 {

 MemTab[last].numAlloc = AllocCount++;

 TotalOccupedMem += NoBytes;

 }

 return MemTab[last++].ptr;

}

MyMemRealloc – realokuje pamięć, alokowaną wcześniej. Przed realokowaniem trzeba odnaleźć w tablice MemTab element, który przechowuje poprzedni adres ptr, wykonać realokowanie, zastąpić poprzedni adres MemTab[…].ptr nowym adresem i poprawić TotalOccupedMem. Parametry AllocCount oraz MemTab[…].numAlloc nie zmieniają się.

MyMemFree – znajduje w tablice MemTab element, który przechowuje poprzedni adres ptr, zwalnia pamięć wywołując funkcje free(…) i kompresuje tablicę MemTab (użyć memmove). Przy tym poprawia TotalOccupedMem. Dekrementuje last.

MyGetOccupedMem() zwraca TotalOccupedMem.