LABORATORIUM 11. DYNAMICZNA ALOKACJA PAMIĘCI. TABLICE DYNAMICZNE.

Cel laboratorium:

Zapoznanie z funkcjami przydziału i zwolnienia pamięci. Zastosowanie tych funkcji do typu tablicowego. Wykorzystanie modułów.

Zakres tematyczny zajęć:

- *tablice statyczne przechowywane w pamięci na stosie,*
- *tablice dynamiczne przechowywane w pamięci na stercie,*
- *dynamiczna alokacja pamięci,*
- *zwalnianie przydzielonej pamięci,*
- *podział programu na moduły.*

Kompendium wiedzy:

Tablice (C89):

- ***statyczna*** - rozmiar tablicy jest znany na etapie kompilacji programu, zmienna przechowywana w pamięci na tzw. stosie (powstaje, gdy program wchodzi do bloku, w którym zmienna jest zadeklarowana, a zwalniana pamięć jest w momencie, kiedy program opuszcza ten blok), liczba elementów tablicy nie ulega zmianie w trakcie działania programu,
- ***dynamiczna*** - rozmiar tablicy jest ustalany na etapie wykonania programu, zmienna przechowywana w pamięci na tzw. stercie (obszar pamięci wspólny dla całego programu, gdzie przechowywane są zmienne, których czas życia nie jest związany z poszczególnymi blokami).

Tablice VLA (*variable length array*) ***(C99):*** rozmiar tablicy może być zdefiniowany w trakcie wykonania programu - może być określony wartością zmiennej, wartość ta nie musi być znana na etapie kompilacji, raz stworzona tablica zachowuje swój rozmiar.

Funkcje dynamicznej alokacji pamięci z biblioteki ***stdlib.h:***

```
void *malloc(size_t n);
void *calloc(size_t n, size_t size);
void * realloc(void *ptr, size_t size);
```

Funkcje przydzielają na stercie wolną pamięć.

Zwracają wskaźnik do **`void`**. Jeśli nie jest możliwe przydzielenie pamięci, to funkcje zwracają wskaźnik **`NULL`**. **`size_t`** jest odwołaniem do pewnego typu danych (typ całkowity bez znaku).

Parametrem funkcji **`malloc`** jest liczba przydzielonych bajtów pamięci.

Parametrami funkcji **`calloc`** są liczba komórek pamięci i rozmiar każdej komórki w bajtach. Pamięć przydzielana funkcją **`calloc`** jest inicjowana zerami.

Funkcja **`realloc`** zmienia rozmiar przydzielonego wcześniej bloku pamięci wskazywanego przez **`ptr`** do rozmiaru **`size`** bajtów.

Bezpośrednie nadanie wartości wskaźnika do **void** wskaźnikowi innego typu nie jest błędem, ale ze względu na czytelność zalecane jest użycie jawnego rzutowania.

```
#include <stdlib.h>

double*tab; int n=10;
tab=(double*)malloc(n*sizeof(double));
if(tab==NULL)
   {printf("Brak wolnej pamieci"); exit(1);}
```

Funkcja **free** zwalnia obszar pamięci alokowany dla wskaźnika będącego jej parametrem

```
void free(void *ptr);
free (tab);
```

Przykład 1. Dynamiczna alokacja pamięci dla tablicy jednowymiarowej:

```
int *t, i;
t = (int *) malloc(10 *sizeof(int));
if (t==NULL)
{printf (" brak wolnej pamięci "); exit(1);}
for (i=0; i<10; i++)
 {*(t + i) = i * i;
  printf("\n%d", *(t + i));
 }
free (t);
```

Funkcja `malloc` *przydziela pamięć dla tablicy 10 liczb całkowitych. Jeśli funkcja zwróci* `null`*, czyli 0, nastąpi wyjście z programu. Jeśli przydzielenie pamięci powiedzie się, to w pętli for będą obliczone i wyświetlone elementy tablicy t: kwadraty liczb od 0 do 9. Następnie pamięć zajęta przez tablicę t będzie zwolniona.*

Przykład 2. Dynamiczna alokacja pamięci dla tablicy dwuwymiarowej:

```
int **p, i;
p =(int*) malloc(sizeof(int *)*5);
for(i=0; i < 5; i++)
p[i] =(int*) malloc(sizeof(int)*10);
```

Deklaracja dynamiczna tablicy p liczb całkowitych składającej się z 5 wierszy i 10 kolumn. Wiersz jest tutaj traktowany jak tablica jednowymiarowa zawierająca 10 liczb, dlatego przydział pamięci wykonany jest w pętli dla każdego wiersza.

Podział programu na moduły:

Moduł- fragment programu kompilowany jako jeden plik składający się z:

- ***interfejsu*** (co plik udostępnia) – pliki nagłówkowe (*.h)
- ***implementacji*** (definicje realizacji) – pliki źródłowe (*.c)

```
module.h
//tu deklaracje funkcji
module.c
#include "module.h"
//tu definicje funkcji
```

Pytania kontrolne:

1. Kiedy należy stosować tablice dynamiczne?
2. Jak dynamicznie alokować pamięć dla tablicy?
3. Jak zwolnić pamięć alokowaną dynamicznie?
4. Jak obsłużyć błąd alokacji pamięci?
5. Jak podzielić program na moduły?
6. Jakie są zalety podziału programu na moduły?

Zadania do analizy

Zadanie 11.1. Tablice dynamiczne jednowymiarowe

- Przeanalizuj przykład programu wykorzystującego tablicę dynamiczną jednowymiarową.
  Funkcja `utworzTD` tworzy jednowymiarowa tablicę dynamiczną o `n` elementach typu `int` i wczytuje dane do tablicy z klawiatury.
  Funkcja `wyswietlTD` wyświetla tablicę `n` elementową liczb całkowitych.
  Funkcja `usunTD` zwalnia pamięć przydzieloną tablicy.
- Podaj tekst w komentarzach.

Main.c

```
1  #include <stdio.h>    //???
2  #include <stdlib.h>
3
4  int *utworzTD(int n); //???
5  void wyswietlTD(int * tabD, int n); //???
6  void usunTD(int *tabD); //???
7  //=====================
8  int main(int argc, char *argv[]) {//???
9    int ile; int *tab1;
10   //tablice dynamiczne
11   printf("Podaj liczbę elementów tablicy ");
12   scanf("%d",&ile);
13   tab1=utworzTD(ile);      //???
14   printf("tablica dyn:\n");
15   wyswietlTD(tab1, ile);   //???
16   usunTD(tab1);            //???
17
18   return 0;
19 }
20 //=====================
21  int *utworzTD(int n)      //???
22  {     int  i;
23   int *tabD = (int*) malloc (n *sizeof(int)); //???
24   if(tabD==NULL)                              //???
25        {printf("Blad przydzialu pamieci\n");
```

```
26          exit(EXIT_FAILURE);}
27
28    for (i=0; i<n; i++)
29    {     printf("wpisz liczbe tab[%d]: ", i);
30          scanf("%d", tabD + i);
31    }
32    return tabD;          //???
33}
34//--------------
35   void wyswietlTD(int * tabD, int n)       //???
36 { int  i;
37    printf ("\nZawartosc tablicy:\n");
38    for (i=0; i<n; i++)
39    printf ("%d\t", *(tabD+i));          //???
40    printf ("\n ");
41}
42//--------------
43   void usunTD(int *tabD)          //???
44   { free(tabD);                   //???
45   tabD=0;
46   }
```

Zadanie 11.2. Tablice dynamiczne jednowymiarowe – podział na moduły

- Przeanalizuj przykład programu wykorzystującego tablicę dynamiczną jednowymiarową z podziałem programu na moduły. Porównaj zawartość i kod projektu z poprzednim zadaniem.
- Podaj tekst w komentarzach.

Main.c

```
//???
1   #include <stdio.h>
2   #include <stdlib.h>
3   #include "TabliceDyn.h"     //???
4
5   int main(int argc, char *argv[]) {
6     int ile; int *tab1;
7     //tablice dynamiczne
8     printf("Podaj liczbę elementów tablicy ");
9     scanf("%d",&ile);
10    tab1=utworzTD(ile);       //???
11    printf("tablica dyn:\n");
12    wyswietlTD(tab1, ile);    //???
13    usunTD(tab1);             //???
14
15    return 0;
16   }
```

TabliceDyn.h //???

```
1  int *utworzTD(int n);      //???
2  void wyswietlTD(int * tabD, int n); //???
3  void usunTD(int *tabD);    //???
```

TabliceDyn.c //???

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include<time.h>
4  #include "TabliceDyn.h"    //???
5  //tablice dynamiczne jednowymiarowe
6  int *utworzTD(int n)                  //???
7  { int  i;
8    int *tabD = (int*) malloc (n *sizeof(int));
9    if(tabD==NULL)
10        {printf("Blad przydzialu pamieci\n");
11         exit(EXIT_FAILURE);
12        }
13   for (i=0; i<n; i++)
14   {    printf("wpisz liczbe tab[%d]: ", i);
15        scanf("%d", tabD + i);
16   }
17   return tabD;
18  }
19  //--------------
20  void wyswietlTD(int * tabD, int n) //???
21 { int  i;
22   printf ("\nZawartosc tablicy:\n");
23   for (i=0; i<n; i++)
24   printf ("%d\t", *(tabD+i));
25   printf ("\n ");
26  }
27  //--------------
28  void usunTD(int *tabD)             //???
29  { free(tabD);
30  tabD=0;
31  }
```

Zadanie 11.3. Tablice dynamiczne dwuwymiarowe

- Przeanalizuj kody funkcji wykorzystujących tablicę dynamiczną dwuwymiarową.

Funkcja `utworzT2D` tworzy dwuwymiarowa tablicę dynamiczną o `n` wierszach i `m` kolumnach elementów typu `int` i wczytuje dane do tablicy z klawiatury.
Funkcja `wyswietlT2D` wyświetla tablicę liczb całkowitych o rozmiarze `n x m`.

- Podaj tekst w komentarzach.

```
    //w jakim pliku umieścić ten kod?
1   int** utworzT2D(int n, int m)        //???
2   {
3     int**tab2D; int i,j;
4     tab2D = (int*) malloc(sizeof(int *)*n); //???
5     if(tab2D==NULL)                          //???
6          {printf("Blad przydzialu pamieci\n");
7           exit(EXIT_FAILURE);
8          }
9     for(i=0; i < n; i++)
10    {
11    tab2D[i] = (int*) malloc(sizeof(int)*m); //???
12     if(tab2D[i]==NULL)                      //???
13         {printf("Blad przydzialu pamieci\n");
14          exit(EXIT_FAILURE);
15         }
16    }
17    for(i=0;i<n;i++)
18    {
19       for(j=0;j<m;j++)
20         {// *(*(tab2D+i)+j)=i*j;     //???
21           printf("tab[%d][%d]= ", i,j);
22           scanf("%d", *(tab2D+i)+j); //???
23         }
24     }
25     return tab2D;          //???
26  }
27  //--------------
28  void wyswietlT2D(int**tab2D,int n, int m)     //???
29  {
30  int i,j;
31   printf("Tablica dynamiczna 2D\n");
32   for(i=0;i<n;i++)
33    {
34        for(j=0;j<m;j++)
35         {
36          printf("%d\t",*(*(tab2D+i)+j)); //???
37         }
38         printf("\n");
39    }
40  }
```

```
//w jakim pliku umieścić ten kod?
1    int**tab2D; int i,j,n,m;           //???
2    printf("\ntablica 2D\n");
3    printf("Podaj liczbę wierszy tablicy ");
4    scanf("%d",&n);
5    printf("Podaj liczbę kolumn tablicy ");
6    scanf("%d",&m);
```

```
7     tab2D = utworzT2D(n,m);          //???
8     wyswietlT2D(tab2D,n,m);          //???
9     usunTD(tab2D);                   //???
```

Zadania do wykonania

Zadanie 11.4. Program z modułem zawierający funkcje na tablicach dynamicznych

Napisz program operujący na tablicach dynamicznych jedno i dwuwymiarowych. Projekt powinien zawierać pliki: `main.c, TabliceDyn.h, TabliceDyn.c`.

W plikach `TabliceDyn.h` i `TabliceDyn.c` umieść deklaracje i definicje funkcji pozwalających na:

- tworzenie, wyświetlenie i usunięcie jednowymiarowej tablicy dynamicznej ***liczb rzeczywistych,***
- tworzenie i wyświetlenie dwuwymiarowej tablicy dynamicznej ***liczb rzeczywistych.***

W pliku `main.c` przetestuj napisane funkcje.

Zadanie 11.5. Tablica statyczna i dynamiczna

Napisz program, który:

- zapełni 100 elementową tablicę statyczną `tabS` wylosowanymi liczbami od 1 do 100 (funkcja `losuj`),
- obliczy, ile z nich jest z podanego przedziału `<a,b>`, a następnie utworzy tablicę dynamiczną odpowiedniego rozmiaru i zapełni ją tymi liczbami (funkcja `nowatabDyn` z parametrami: wskaźnik do tablicy statycznej `tabS`, liczba jej elementów `n`, przedziały `a` i `b`, wskaźnik do liczby elementów tablicy dynamicznej `m`; funkcja zwraca wskaźnik do tablicy dynamicznej).

Wyświetl obydwie tablice. Wykorzystaj odpowiednie funkcje.

Zadanie 11.6. Tworzenie tablic dynamicznych z tablicy dwuwymiarowej

Dane: `tab2` - tablica liczb rzeczywistych o wymiarach `n` wierszy, `m` kolumn.
Napisz funkcje:

- `f1` – tworzy tablicę `tabD`, zawierającą elementy dodatnie,
- `f2` – tworzy tablicę `tabU`, zawierającą elementy ujemne.

Napisz program, w którym wczytane są dane, wywołane funkcje, wyświetlone wyniki.

Zadania dodatkowe

Zadanie 11.7. Maraton

Napisz program ewidencjonujący czasy osiągnięte przez zawodników maratonu i wyszukujący zwycięzcę. Liczba zawodników `n` nie jest dokładnie znana, zakłada się, że startowa pula numerów jest ograniczona do 300 osób. Należy ewidencjować dokładnie tyle

czasów ile zawodników. Program powinien wyszukać najlepszy wynik i wyświetlić numer startowy/ numery osoby/ osób z tym wynikiem. Wykorzystaj odpowiednie funkcje.

Zadanie 11.8. Obliczenie sum w wierszach i kolumnach tablicy dwuwymiarowej

Napisz funkcję obliczającą sumy w wierszach i sumy w kolumnach w dwuwymiarowej tablicy liczb rzeczywistych i zwracającą wyniki, jako dwie tablice. Rozmiary (`n, m`) i elementy danej tablicy `tab` są parametrami przekazanymi do funkcji. Tablica sum w kolumnach `sumaK` jest przekazana z funkcji jako parametr wskaźnikowy, wskaźnik do utworzonej tablicy sum w wierszach `sumaW` jest przekazany poprzez return.

Prototyp funkcji może wyglądać następująco:

```
float * sumaW(int n, int m, float tab[][m], float * sumaK);
```

Napisz program, w którym wczytane są dane, wywołana funkcja, wyświetlone wyniki.

Zadanie 11.9. Tworzenie tablicy dynamicznej dwuwymiarowej

W hurtowni jest `n` towarów. Tablica `dane` zawiera w kolumnie zerowej ceny, w kolumnie pierwszej ilości towarów. Napisz funkcję tworzącą tablicę informacji o towarach (cena, ilość), których wartość jest większa od liczby podanej przez użytkownika. Tablice są deklarowane dynamicznie.

Napisz program, w którym wczytane są dane, wywołana funkcja, wyświetlone wyniki.

Zadanie 11.10. Tablica dynamiczna zwracana przez wskaźnik

Tablica `punkty` zawiera współrzędne `n` punktów na płaszczyźnie. Napisz funkcję tworzącą tablicę odległości punktów od początku układu współrzędnych. Wskaźnik do tablicy jest przekazany z funkcji przez return.

Napisz program, w którym wczytane są dane, wywołana funkcja, wyświetlone wyniki.

Zadanie 11.11. Moduł z funkcjami do obsługi tablic

Utwórz funkcje wykonujące podstawowe operacje na tablicach jedno i dwuwymiarowych o podanych rozmiarach i zapisz je w module `tablice.h`. Parametrami funkcji powinny być rozmiary i tablica. Podstawowe operacje (oprócz wczytania i wyświetlenia zawartości) to:

- sumowanie wszystkich elementów tablicy,
- obliczanie średniej arytmetycznej elementów tablicy,
- szukanie wartości minimalnej i maksymalnej,
- zliczanie elementów o podanej wartości,
- wykonanie operacji na wybranych elementach tablicy.

Przetestuj te funkcje na utworzonych tablicach dynamicznych.