

## Dynamiczna alokacja pamięci

Do tej pory używaliśmy tablic, które miały z góry zadany rozmiar. Deklarowaliśmy je w bardzo prosty sposób:

```
double x[100];
```

Oczywiście chcielibyśmy poszerzyć funkcjonalność naszego programu tak, aby była możliwa deklaracja tablic o długości określonej już w trakcie działania programu. Mówimy wówczas o tzw. dynamicznej (tzn. wykonywanej podczas działania programu, a nie podczas kompilacji) alokacji pamięci. Taki mechanizm jest jak najbardziej dopuszczalny w języku C, składa się na niego kilka etapów:

- Deklaracja wskaźnika do nowotworzonej tablicy.
- Alokacja pamięci dla danej tablicy - używa się w tym celu funkcji `malloc()`.
- Zwolnienie pamięci - funkcja `free()`.

Nowe funkcje wymagają użycia biblioteki `stdlib.h`. Poniższy kawałek kodu ilustruje cały mechanizm:

```
#include <stdlib.h>

void main()
{
    int N=100;
    double *x;           // wskaźnik do pierwszego elementu tablicy

    x = (double*)malloc(N*sizeof(double)); // alokacja

    free(x);             // zwolnienie pamięci
}
```

Kluczowym elementem całego mechanizmu jest właściwe zastosowanie funkcji `malloc`. Jej argumentem jest rozmiar pamięci, o którą wnioskujemy. Potrzebujemy  $N$  razy rozmiar zajmowany przez zmienną typu `double` - stąd obecność funkcji `sizeof()`, która zwraca rozmiar danego typu zmiennych. Ponadto używamy mechanizmu rzutowania - przed funkcją `malloc` pojawia się rzutowanie na typ `double` poprzez wywołanie `(double*)`<sup>1</sup>. Po pracy na tablicy należy pamiętać o zwolnieniu pamięci, którą ona zajmowała. Stąd funkcja `free()`.

<sup>1</sup>programując w czystym C można by teoretycznie zrezygnować z rzutowania, wówczas `malloc`

## Wczytywanie i rysowanie konturów

Nowy mechanizm będzie nam potrzebny do wykonania zadania polegającego na wczytaniu współrzędnych konturów map z plików oraz narysowaniu tych map na ekranie. Współrzędne konturów dwóch map są zapisane w dwóch plikach, `plik1.txt` (do [ściągnięcia tu](#)) oraz `plik2.txt` (do [ściągnięcia tu](#)). Pliki mają następującą strukturę:

```
156
12.67    768.3254
14.98    768.3254
17.462   766.51075
...
```

Pierwsza linia pliku zawiera ilość punktów zapisanych w pliku. Poniżej współrzędne  $x$  oraz  $y$  są zapisane w oddzielnych kolumnach.

## Ćwiczenia

- Otwórz oba pliki w programie głównym.
- Wczytaj pierwsze linie obu plików do programu. Wydrukuj na ekran te wartości, aby upewnić się, że wykonano ten podpunkt bezbłędnie.
- Użyj wczytanych wartości do alokacji tablic, które będą przechowywać współrzędne punktów. Użyj mechanizmu alokacji dynamicznej.
- Wczytaj współrzędne z obu plików do wcześniej zadeklarowanych tablic. Upewnij się, czy wykonano to dobrze.
- Napisz funkcję `void PrintCoords(double *x, double *y, int N, int start, int end)`, która będzie drukować na ekranie zadany przedział współrzędnych z danej mapy. Funkcja ma informować, gdy zadany przedział nie może być wydrukowany (bo np. kres górny przekracza długość tablicy)
- napisz funkcję `void Display()`, która będzie rysować zadany kontur mapy na ekranie w oknie graficznym. Kontur ma być narysowany za pomocą linii, które będą łączyć kolejne punkty. Zwroć uwagę na to, by połączyć również ostatni punkt z pierwszym. Wyświetl oba kontury na jednym obrazku. Co to za mapy?

zwróci wskaźnik typu `void*`. Jednak powszechnie stosujemy kompilatory języka C++, który jest językiem o tzw. silnej kontroli typów i nie zezwala na to, by wskaźnik nie miał typu. Nie zagłębiając się bardziej w szczegóły, należy po prostu wyrobić w sobie nawyk rzutowania przed użyciem funkcji `malloc()`, aby uniknąć wielu nieprzyjemnych sytuacji w przyszłości.

## Trochę geografii

Mając już kontur, możemy policzyć kilka interesujących rzeczy:

### Ćwiczenia

- Napisz funkcję `double Perimeter()`, która zwróci obwód zadanego konturu.
- Napisz funkcję `double Area()`, która policzy oraz zwróci powierzchnię danej mapy. Powierzchnię mapy należy policzyć jako sumę powierzchni trójkątów, których podstawy są kolejnymi odcinkami konturu a wierzchołek jest zlokalizowany gdziekolwiek (najwygodniej położyć go w zerze). Pole takiego trójkąta będzie po prostu połową modułu iloczynu wektorowego, gdzie wektorami są boki trójkąta:

$$S_i = \frac{1}{2} |\mathbf{v}_i \times \mathbf{v}_{i+1}| = \frac{1}{2} (x_i y_{i+1} - x_{i+1} y_i)$$

Pamiętaj o ostatnim i pierwszym punkcie. Zwróć uwagę, że otrzymane pole może być ujemne. Od czego to zależy?

- Policz rozciągłość południkową i równoleżnikową obu obszarów. Zaznacz najbardziej skrajne punkty na obu mapach za pomocą kółka.