

Dynamiczna alokacja pamięci

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

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

lub też:

```
const int N = 100;
double x[N];
```

Chcielibyśmy poszerzyć funkcjonalność naszego programu tak, aby była możliwa deklaracja tablic o długości określonej dopiero w trakcie działania programu. Mówimy wówczas o tzw. dynamicznej (tzn. wykonywanej podczas działania programu, a nie podczas kompilacji) alokacji pamięci. W języku C, na taki mechanizm składa się kilka etapów:

- Deklaracja wskaźnika, który będzie zawierał adres tworzonej 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 fragment kodu ilustruje cały mechanizm:

```
#include <stdlib.h>

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

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

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

Kluczowym elementem mechanizmu jest właściwe zastosowanie funkcji `malloc()`. Jej argumentem jest rozmiar pamięci, o którą wnioskujemy. Potrzebujemy rozmiaru równego N razy rozmiar zajmowany przez zmienną typu `double`. Stąd wynika

obecność funkcji `sizeof()`, która zwraca rozmiar danego typu zmiennych. Ponadto używamy mechanizmu rzutowania – przed funkcją `malloc()` pojawia się rzutowanie na wskaźnik do typu. W naszym przypadku jest to wywołanie `(double *)`¹. Po pracy na tablicy, należy pamiętać o zwolnieniu pamięci, którą ona zajmowała. Służy do tego funkcja `free()`.

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 plikach: [plik1.txt](#) oraz [plik2.txt](#). Pliki mają następującą strukturę:

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

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

Ćwiczenia

1. Otwórz oba pliki w programie głównym.
2. Wczytaj pierwsze linie obu plików do programu. Wydrukuj na ekran te wartości, aby upewnić się, że wykonano ten podpunkt bezbłędnie.
3. Użyj wczytanych wartości w celu dynamicznej alokacji tablic, które będą przechowywać współrzędne punktów.
4. Wczytaj współrzędne z obu plików do wcześniej zadeklarowanych tablic. Upewnij się, że dane wczytano dobrze.
5. Napisz funkcję `void printCoords(double *x, double *y, int N, int start, int end)`, która będzie drukować na ekran zadany przedział

¹programując w C można by zrezygnować z rzutowania, wówczas `malloc()` 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 w szczegóły, należy wyrobić sobie nawyk rzutowania wskaźnika zwracanego przez funkcję `malloc()`, aby uniknąć nieprzyjemnych sytuacji w przyszłości.

- współrzędnych z danej mapy. Funkcja ma informować, gdy zadany przedział nie może być wydrukowany (bo np. zmienna `end` przekracza długość tablicy).
6. 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. Zwróć uwagę na to, aby połączyć również ostatni punkt z pierwszym. Wyświetl oba kontury na jednym obrazku. Co to za mapy?

Trochę geografii

Mając kontur, możemy policzyć kilka interesujących rzeczy:

Ćwiczenia

7. Napisz funkcję `double perimeter()`, która zwraca obwód zadanego konturu.
8. Napisz funkcję `double area()`, która zwraca 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?

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