INFORMATYKA I: INSTRUKCJA 5

Tablice

Celem zajęć jest wprowadzenie do używania tablic w języku C. Tablicą (ang. *array*) nazywamy ciąg zmiennych zgromadzony pod jedną globalną nazwą, które są identyfikowane indeksami. Na tych zajęciach zajmiemy się tylko tablicami statycznymi tzn. takimi, których rozmiar jest określany w momencie deklaracji[1]. Tablicę statyczną deklarujemy tak, jak zwykłą zmienną, przy czym dodatkowo określamy jej długość. Wszystko wygląda, jak w przykładowym kodzie poniżej:

```
double a[4];     // deklaracja tablicy

a[0] = 5.5;    // przypisanie wartości do zmiennych
a[1] = 3.521;
a[2] = 6.45;
a[3] = 4.51;
```

Zwróć uwagę, że elementy tablicy sa indeksowane od 0 do $n-1$, gdzie n to rozmiar tablicy. Można również zainicjalizować wszystkie elementy tablicy natychmiast (taki mechanizm jest użyteczny, jeśli wektory są stosunkowo krótkie):

```
double b[3] = { 1.2, 2.4, -4.3};
```

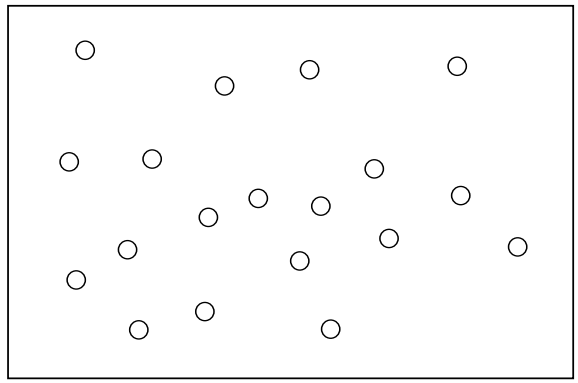
Piłki

Zadanie polegać będzie na wygenerowaniu zestawu piłek w oknie graficznym oraz na wyszukiwaniu i określaniu ich specyficznych cech, jak np. minimalna, średnia i maksymalna pozycja, maksymanla masa itp. Przykładowy zbiór takich piłek jest widoczny na Rysunku 1.

1 Inicjalizacja

Nasze piłki będą przychowywane tylko jako zestawy współrzędnych, ich prędkości oraz masy. Gdy będziemy chcieli obejrzeć piłki w oknie graficznym, po

[1] bardziej zaawansowany mechanizm alokacji tablic będzie tematem następnych zajęć



Rysunek 1: Piłki w ramce.

prostu użyjemy funkcji `circle`. Toteż w symulacji będą potrzebne następujące wektory[2]:

```
double x[10],y[10];            // współrzędne piłek
double vx[10], vy[10];         // składowe prędkości piłek
double m[10];                 //masy piłek
```

Pętla `for`

Większość operacji na tych zmiennych będziemy wykonywać, używając funkcji, które będą przyjmować wprowadzone wyżej wektory jako argumenty. Funkcje będą musiały mieć podaną długość wektorów tak, aby można było wykonać pewne operacje dla każdego z elementów tego wektora. Jeśli chcemy np. zainicjalizować wszystkie współrzędne wartością 0, piszemy funkcję następującej treści:

```
void init(double *x, double *y, int N){

      for ( int i=0; i < N; ++i){
          x[i] = 0.0;
          y[i] = 0.0;}
}
```

[2] tablice często będziemy nazywać wektorami, ze względu na fakt, że określenie "tablica"kojarzy się z obiektem o większej ilości wymiarów np. z macierzą

Wykorzystaliśmy tutaj pętlę `for`, która pobiera 3 argumenty:

- wartość startową,
- warunek działania (pętla działa, dopóki warunek $i < N$ jest spełniony),
- operację na argumencie (tutaj zwiększamy i o 1, co będzie najpowszechniejszą praktyką[3])

Taką funkcję wywołujemy w programie głównym, podając nazwy wektorów, na których ma ona działać oraz długość tych wektorów:

```
init(x, y, 10);
```

Zauważmy, że funkcja `init` pobiera 2 wskaźniki do wektorów(`x` oraz `y`) oraz jedną wartość (10). Dzięki temu funkcja operuje bezpośrednio na wektorach, na których ma operować i niczego nie musi zwracać[4].

Uwaga

Ponieważ `x` oraz `y` są wskaźnikami do pierwszych elementów tablic, można użyć mechanizmu wyłuskania wartości ze wskaźników i iterować się po wskaźnikach. Poniższy fragment kodu pokazuje dwa równoważne sposoby dostępu do wartości z tablicy:

```
double a[3];
// po wartościach:
a[0] = 1.2;         a[1] = 3.13;         a[2] = 0.22;
//albo na wskaźnikach:
*(a)   = 1.2;           *(a+1) = 3.13;        *(a+2) = 0.22;
```

Ćwiczenia

Przed wykonaniem ćwiczeń upewnij się, że załączono bibliotekę `winbgi2.h`, gdyż będziemy korzystać z grafiki.

1. Zadeklaruj wymienione wyżej wektory położeń, prędkości oraz mas. Utwórz 20 piłek.
2. Zadeklaruj okno graficzne o wymiarach $Lx \times Ly$.
3. Napisz funkcję `init(double *x, double *y, double *vx, double *vy)`, która wylosuje współrzędne położeń początkowych tak, aby powstałe piłki mieściły się w oknie graficznym a składowe prędkosci zawierały się w przedziale $\langle -20, 20 \rangle$. Użyj funkcji `rand()` znanej z poprzednich zajęć.
4. Wypisz na ekran współrzędne wszytkich piłek oraz ich prędkości.
5. Napisz funkcję `display(double *x, double *y, int N)`, która wyświetli położenie i prędkości piłek. Wyświetl piłki.
6. Każdej piłce przypisz masę - niech piłka o indeksie i ma masę $m_i = 2i^2+1$.

2 Analiza położeń i mas

Często nasz zestaw danych musimy poddać jakiejś analizie. Np. chcielibyśmy wiedzieć, która piłka ma największą współrzędną y. W tym celu musimy dokonać przeszukiwania w danym zbiorze danych. Ponadto chcielibyśmy wykonać taką operację możliwie niskim kosztem. W tym przypadku możemy zrobić to, iterując się tylko raz po całym zbiorze piłeczek. Dodatkowo utworzymy tylko tymczasową zmienną (bufor), która będzie przechowywać maksymalna wartość współrzędnej y. Ten bufor nazwiemy `ymax` , gdyż po działaniu pętli to w nim zostanie wartość maksymalna. To wszystko ilustruje poniższy kawałek kodu:

```
double ymax=0.0; //od czegoś trzeba zacząć

for( i=0; i < N; ++i){
     if(y[i] > ymax){
          ymax=y[i];}
}
```

Powyższa pętla kroczy po wszystkich współrzędnych y i jeśli któraś z nich jest większa od wartości aktualnie znajdującej się w buforze, jej y staje się nowym maksimum. Nie trudno zauważyć, że jeden taki cykl załatwia sprawę do końca. Oczywiście w tej samej pętli moglibyśmy zrobić inne interesujące nas rzeczy. Np. możemy zabrać się za liczenie średniej masy piłek:

```
double m_avg = 0.0;;
```

[3] Teoretycznie możemy w tym miejscu wykonać dowolną operację, jednak dla czytelności kodu zazwyczaj zwiększamy licznik pętli

[4] Zasady działania na wskaźnikach opisano w Instrukcji 4.2.

```
for( i=0; i < N; ++i){
        m_avg+=m[i]; //liczymy całkowitą masę piłek
}

m_avg = m_ang/N; //liczymy średnią
```

Ćwiczenia

1. Napisz fragment kodu, który znajduje piłki o minimalnej i maksymalnej współrzędnej x oraz y. Na ekranie wypisz te minima i maksima wraz z indeksami tych piłek.
2. Napisz funkcję `crossOut(double *x, double *y, int i)`, która przekreśli krzyżykiem piłkę o indeksie i. Do skreślania użyj dwóch funkcji `line`. Za pomocą tej funkcji skreśl piłki z poprzedniego podpunktu.
3. Napisz fragment kodu, który policzy minimalną, średnią i maksymalną energię kinetyczną piłek. Wypisz te wartosci na ekran. Energię kinetyczną liczymy ze wzoru $E_k = 1/2m(v_x^2 + v_y^2)$.
4. Na środku ekranu narysuj koło o promieniu równym 0.3 przekątnej okna graficznego. Następnie wykreśl wszystkie piłki, których środki nie znajdują się w obszarze tego koła.
5. Napisz funkcje `swapX` i `swapY`, które zamienią wskazane współrzędne piłek w zbiorze wg schematu pierwsza z ostatnią, druga z przedostatnią itd. Zamiana powinna odbyć się w obrębie jednej petli! Wyświetl nowy zbiór piłek po każdej zamianie.

3 *Zaawansowane przeszukiwanie

Do tej pory mieliśmy do czynienia z iteracjami, których liczba rosła liniowo (tzn. proporcjonalnie) do rozmiaru zbioru. Co gdybyśmy chcieli np. porównywać piłki między sobą? Wówczas musielibyśmy iterować się po wszystkich parach piłek. Np. jeśli piłek jest 10, par jest 45. Taki proces ma koszt kwadratowy (dokładnie $n(n-1)/2$). W poniższym przykładzie znaleźć maksymalną odległość między dwoma piłkami. Odległość między dwoma piłkami będzie dana wzorem:

$$L = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

W najprotszej wersji możemy najpierw stworzyć tablicę, która przechowa wszystkie możliwe odległości, a następnie znaleźć jej maksimum. Jednak należy zrobić to bardziej elgancko, bez deklarowania dodatkowych tablic. Załatwi to następujący kod:

```
double L_max=0.0;
double L_tmp;        //dodatkowa zmienna dla czytelności kodu

for(i =0; i < 10; +i){
     for(j=i+1; j < 10; ++j){
          L = sqrt((x[i]-x[j])*(x[i]-x[j])
                  + (y[i]-y[j])* (y[i]-y[j]));
          if(L>L_max){
                   L_max = L;}
          }
}
```

Zauważmy, że podpętla po indeksie j ma zakres zależny od i: nie ma sensu przeszukiwać wszystkich piłek wstecz. Wystarczy, że każda piłka o indeksie i policzy swoją odległość do piłek, których indeksy następują po niej samej.

Ćwiczenia

1. Znajdź najmniejszą i najwiekszą odległość między piłkami. Wypisz te odległości oraz indeksy tych piłek na ekran.
2. Narysuj linie łączące 2 najbliższe piłki i 2 najdalsze.
3. Połącz liniami piłki, które są od siebie dalej niż $Lx/2$.