INFORMATYKA I: PROJEKTY DOMOWE

1 Gra w bilard

Napisz program, który będzie odzwierciedlał kawałek gry w bilard. Na prostokątnej planszy losuj położenia kilku bil. Jedna z nich powinna być w dowolny sposób wyróżniona (ma reprezentować białą bilę). Użytkownik ma mieć możliwość w dowolny sposób (przez wpisanie składowych z klawiatury lub obsługę myszki; w pliku `winbgi.h` możesz odszukać przydatne funkcje) zadania kierunku, w jakim rozpędzi tę bilę. Będzie to odpowiadać uderzeniu kijem białej bili. Następnie symuluj ruch bili i zderzenia między nimi oraz ściankami i sprawdzaj, czy któraś z rozpędzonych bil z odpowiednią dokładnością trafi w róg stołu (wykrywaj takie zdarzenie - powinna wtedy znikać). Odpowiednie wzory opisujące zderzenia dwóch bil znajdziesz w Instrukcji 5.2.

2 Wykres

Napisz, korzystając z używanej na laboratorium biblioteki `winbgi2`, funkcję do rysowania wykresów. Funkcja powinna jako argumenty przyjmować wskaźniki do dwóch tablic (odpowiednio odciętych i rzędnych) oraz zmienną typu całkowitego określającą rozmiar tych tablic. Zadbaj o to, aby wykres był rysowany na ekranie linią ciągłą, odpowiednio generowane osie rzędnych i odciętych oraz zaznaczone wartości na tych osiach (możesz do tego użyć odpowiedniej funkcji biblioteki `winbgi2`, która drukuje tekst do okna graficznego - znajdziesz jej nazwę w pliku nagłówkowym). Możesz również dodać możliwość przekazania do funkcji nazw (etykiet) obu osi. Przetestuj swoją funkcję dla kilku różnych zestawów danych.

3 Burzenie murów zamku

Napisz program, który będzie stanowił modyfikację program z Laboratorium 9. Tym razem stwórz geometrię zamku z pojedynczych pikseli. Napisz odpowiednie funkcje, które stworzą zamek piksel po pikselu (najwygodnie napisać sobie dwie funkcje, które z pikseli będą potrafiły tworzyć prostokąt i trójkąt, a następnie z tych kształtów wygenerować zamek). Następnie program ma realizować ten sam algorytm ustawiania armaty i oddawania strzału, jednak w momencie, w którym kula znajdzie się na dowolnym pikselu należącym do zamku, zakończy swój lot i wytworzy w tym miejscu wyrwę w murze (tzn. usunie wszystkie piksele w promieniu np. 10 pikseli od miejsca uderzenia). Promień wyrwy uzależnij (np. liniowo) od prędkości, z jaką kula uderza w mur.

4 Kółko i krzyżyk

Napisz program, który będzie z użytkownikiem grał w kółko i krzyżyk. Plansza ma być rysowana w oknie graficznym, komunikacja z użytkownikiem ma być prowadzona za pośrednictwem klawiatury. Opracuj dwa różne algorytmy, według których program będzie wybierał kolejne pola do postawienia swojego markera. Najprostszy algorytm ma być algorytmem losowym, więc komputer ma bez zastanawiania się nad ruchami losować dowolne, wolne miejsce do postawienia swojego znaku. Program ma wykrywać zwycięstwo, przegraną bądź remis. Drugi algorytm powinien zawierać analizę, czy dany ruch ma sens. Możesz np. analizować wszystkie możliwe ruchy wprzód i wybierać takie pole, które będzie dawało największe szanse wygranej. Możesz też opracować swój autorski algorytm lub spróbować zaczerpnąć wiedzy na temat podstawowych, powszechnie stosowanych algorytmów gry w kółko i krzyżyk.

5 Gra w statki

Napisz program, który będzie grał z użytkownikiem w statki. Na poczatku ma losować położenia statków na swojej planszy (pamiętamy, że statki nie mogą się stykać ani rogiem ani bokiem) oraz pozwolić użytkownikowi wybrać miejsca, w których chce ustawić swoje statki. Następnie ma przeprowadzić grę w statki (zastanów się nad możliwie najsprytniejszym sposobem, w jaki komputer ma ostrzeliwać pole gracza) i wykrywać wygraną.

6 Liczenie całki metodą Monte Carlo

Napisz program liczący przybliżoną wartość całki $\int_a^b f(x)dx$ metodą Monte Carlo. Załóż, że wartości funkcji podcałkowej są dodatnie. Metoda Monte Carlo obliczania całek oznaczonych działa w następujący sposób: Znajdź pole prostokąta zawierającego wykres funkcji; generuj w sposób losowy punkty z tego prostokąta. Przybliżona wartość całki jest równa polu prostokąta pomnożonemu

przez ułamek określający, ile z wylosowanych punktów trafiło w obszar pod wykresem funkcji.

7 Kalkulator pochodnych

Napisz program, który drukuje na ekranie wzór na pierwszą pochodną funkcji zadanej w pliku. Wzór funkcji w pliku jest postaci $f(x) * g(x)$, gdzie funkcje $f(x)$ i $g(x)$ to jedno z wyrażeń typu x^n, $sin(a * x)$ lub $cos(b * x)$. Przykładowy wzór podany w pliku tekstowym to:

```
x^7*cos(3.14*x)
```

8 Błądzenie losowe

Napisz program, który będzie wykonywał błądzenie losowe sporej grupy ludzi (np. 10000 osób) startujących z jednego punktu. Każda osoba wykonuje w każdym kroku czasowym przesunięcie w lewo albo w prawo z zadanym prawdopodobieństwem. Utwórz histogram, na którym bedzie widać, jak daleko zaszło ile osób. Przetestuj to dla różnych prawdopodobieństw wykonywania kroku w daną stronę. Dopuszczale są tylko dwa ruchy: krok w lewo albo krok w prawo. Nie ma możliwosci pozostanie w miejscu.

9 Szyfrowanie i deszyfrowanie

Napisz program, który pozwoli na zaszyfrowanie danego kawałka tekstu oraz złamanie danego szyfru. Zasada szyfrowania jest następująca: Wszystkie litery wiadomości nieszyforwanej mają zostać przesunięte o pewną stałą odległość. Np. dla przesunięcia 3 będzie to wyglądało tak:

```
ALA MA KOTA -> DOD PD NRXD
```

Łamanie szyfru będzie się odbywać w oparciu o fakt, że łamiący szyfr będzie posiadał bazę słów, spośród których któreś na pewno znajduje się w tekście. Dzięki temu, próbując wszytkich możliwości dekryptażu (a jest ich tyle, ile liter w alfabecie łacińskim), wybierze poprawną przez sprawdzenie czy znajduje się tam dane słowo. Oczywiście taki algorytm nie jest do końca jednoznaczny, jednak powinien działać rozsądnie przy krótkich wiadomościach. Wiadomość oryginalna, szyfrowana oraz baza danych słów-kluczy mają być wczytywane z pliku.

10 Histogram z zadaną podziałką

Należy napisać program, kóry będzie rysować histogram z zadaną podziałką w oparciu o dostarczony zbiór danych z pliku. Podziałka histogramu oznacza na ile sektorów należy podzielić wykres. Należy samemu zdobyć/wygenerować dane źródłowe (np. poprzez losowanie).

11 Działania na macierzach

Należy napisać program, który wykonuje działania algebraiczne na macierzach: dodaje, odejmuje oraz mnoży je przez siebie. Macierze mają być wczytywane z plików, program ma sprawdzać, czy ich wymiary są odpowiednie do wykonania danych operacji, a następnie generować plik z wynikiem.

12 Gra w węża

Należy napisać program, który bedzie realizował grę w węża w oknie graficznym biblioteki `winbgi.h`. W najprostszej wersji wąż może w każdym kroku czasowym upominać się o dane z klawiatury, w wersji zaawansowanej wąż porusza się, a kierunki są przyjmowane w trakcie działania programu. Dane kierunku mogą być wprowadzane za pomocą myszki (funkcje `mouseclickx(), mouseclicky()` albo klawiatury (funkcja `getch()` i pokrewne). Wąż oczywiście rośnie po zdobyciu pokarmu oraz umiera po zderzeniu sam ze sobą albo ze ścianą. Uwaga: interakcja z danymi z klawiatury staje sie możliwa dopiero po kliknięciu myszą na działające okno grafiki i pozostawieniu kursora w tym obszarze.

13 Gra w życie

Należy napisać program symulujący grę w życie wg klasycznych zasad Conway'a (patrz `Wikipedia`). Program ma wyświetlać stan populacji w każdym kroku używając biblioteki `winbgi.h`

14 Płonący budynek

Należy napisać program, który symuluje ucieczkę grupy ludzi z płonącego pomieszczenia: w momencie wybuchu pożaru ludzie (kółka w oknie graficznym)

zaczynają się kierować w stronę wyjścia. Niestety biegną na oślep i mogą się ze sobą zderzyć (odbijają się wg schematu z Laboratorium 5.2), przez co przez chwilę pruszają się bezwładnie, ale po kilku krokach orientują się w przestrzeni i znów nadają sobie prędkość w kierunku wyjścia. Może potem dojść do następnych zderzeń. Zbadaj, jak zależy możliwość ucieczki takiej grupy od rozmiarów pojedynczego człowieka oraz od ilości ludzi wewnątrz. Wyjście jest jedno, kółko po prostu znika, gdy do niego dotrze.

15 Totolotek** **(wymaga sporo pracy, sugeruje się je grupom dwuosobowym)**

Należy napisać program symulujący maszynę do losowania używaną w Totolotku. Program w dużej mierze opiera się na Laboratorium 5.2 (zderzenia ze ścianami, kolizje), lecz należy uwzględnić działanie pól siłowych: grawitacji (która po prostu powinna zmniejszać składową y prędkości w każdym kroku funkcji `run`) oraz dmuchawy (w pewnym obszarze okna graficznego piłki powinny być przyspieszane w pewien sposób - tutaj jest miejsce na inwencję użytkownika). Dodatkowo należy dodać niewielką ilość dyssypacji energii do programu np. przy kazdym zderzeniu niech piłki zachowują tylko 95% energii kinetycznej. Wreszcie konieczna jest współpraca z myszką/klawiaturą: na pierwsze kilknięcie następuje zwolnienie blokady (piłki wpadają do maszyny), na drugie zaczyna się dmuchanie, na trzecie następuja zassanie piłek do boksu. Należy pamiętać, że w boksie może być tylko jedna piłka. Oczywiscie piłki powinny się ze sobą zderzać. Należy z miarę możliwości wprowadzić lekkie zaburzenia na początku działani programu (losowo rozmieścić piłki w boksie), by wynik był niedeterministyczny.