Projekty Domowe

Większość zadań wymaga skorzystania z biblioteki `winbgi2`. Niektóre zadania wymagają zaimplementowania obsługi myszki lub klawiatury w celu sterowania przebiegiem programu. Niezbędne funkcje można znaleźć także w bibliotece `winbgi`. Dodatkowo, na końcu niniejszej instrukcji umieszczono opisy podstawowych funkcji służących do obsługi myszki i klawiatury.

1. Gra w bilard

Napisz program, który będzie symulował uproszczoną grę w bilard.
Na prostokątnej planszy wylosuj 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) zadania kierunku, w jakim rozpędzi tą bilę. Będzie to odpowiadać uderzeniu kijem białej bili. Następnie zasymuluj ruch bili oraz zderzenia miedzy bilami oraz ściankami. Sprawdzaj, czy któraś z rozpędzonych bil, z odpowiednia dokładnością, trafi w róg stołu (wykrywaj takie zdarzenie – bila powinna wtedy znikać).
Odpowiednie wzory opisujące zderzenia dwóch bil znajdziesz w Instrukcji 5.

2. Wykres

Napisz program, który będzie rysował wykresy funkcji matematycznych. Program powinien zawierać funkcję rysującą, która jako argumenty będzie 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 linia ciągłą, osie rzędnych i odciętych były odpowiednio generowane a na osiach były zaznaczone wartości (możesz do tego użyć funkcji `outtextxy()` z biblioteki `winbgi2`, która drukuje zadany tekst w miejscu o zadanych współrzędnych). Możesz również dodać możliwość przekazania do funkcji nazw (etykiet) obu osi. Przetestuj swoja funkcje dla kilku rożnych zestawów danych.

3. Burzenie zamku

Napisz program, który będzie stanowił modyfikacje program opisanego w Instukcji 9. Napisz odpowiednie funkcje, które stworzą zamek piksel po pikselu (najwygodniej 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. Gra w kółko i krzyżyk

Napisz program, który będzie grał z użytkownikiem w kółko i krzyżyk. Plansza ma być rysowana w oknie graficznym, a komunikacja z użytkownikiem ma być prowadzona za pośrednictwem klawiatury lub myszki. Opracuj dwa rożne algorytmy, według których program będzie wybierał kolejne pola, w których postawi swój marker. Najprostszy algorytm ma być algorytmem losowym – 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 w przód 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 początku program ma losować położenia statków na swojej planszy (pamiętaj, że statki nie mogą się stykać ani rogiem ani bokiem). Następnie użytkownik ma wybrać miejsca, w których chce ustawić swoje statki (może tego dokonać za pomocą myszki lub klawiatury). Ostatecznie, program 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, który wyznacza przybliżoną wartość całki $R_b = \int_a^b f(x)dx$ metodą Monte Carlo. Załóż, ze 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,
- oblicz przybliżoną wartość całki jako pole prostokąta pomnożone przez ułamek określający, ile z wylosowanych punktów trafiło w obszar pod wykresem funkcji.

Wybierz jakąś funkcję i zilustruj funkcję oraz punkty na wykresie (por. Instukcja 4).

7. Kalkulator pochodnych

Napisz program, który drukuje na ekranie wzór na pierwszą pochodną funkcji zadanej w pliku. Załóż, że wzór funkcji w pliku jest postaci $f(x) \cdot g(x)$, gdzie funkcje $f(x)$ i $g(x)$ to jedno z wyrażeń typu x^n, $\sin(a \cdot x)$ lub $cos(b \cdot x)$. Przykładowy wzór podany w pliku tekstowym to: $x^7 \cdot \cos(\pi \cdot x)$.

8. Błądzenie losowe

Napisz program, który będzie symulował błądzenie losowe sporej grupy ludzi (np. 10000 osób) startujących z jednego punktu. Każda osoba wykonuje w każdym kroku czasowym przesuniecie w lewo lub w prawo z zadanym prawdopodobieństwem. Utwórz histogram liczby osób od odległości, na którą zaszli. Przetestuj program dla rożnych prawdopodobieństw wykonywania kroku w dana stronę.
Dopuszczalne są tylko dwa ruchy: krok w lewo albo krok w prawo. Nie ma możliwości pozostanie w miejscu.

9. Szyfrowanie i deszyfrowanie

Napisz program, który pozwoli na zaszyfrowanie danego fragmentu tekstu oraz złamanie danego szyfru. Zasada szyfrowania jest następująca: wszystkie litery wiadomości niezaszyfrowanej mają zostać przesunięte o pewną stałą odległość. Np. dla przesunięcia o 3 będzie to wyglądało tak: $ALAMAKOTA- > DODPDNRXD$.
Łamanie szyfru będzie się odbywać w oparciu o fakt, że funkcja łamiąca szyfr będzie posiadała bazę słów, spośród których któreś na pewno znajduje się w tekście. Dzięki temu, próbując wszystkich możliwości dekryptażu (a jest ich tyle, ile liter w alfabecie łacińskim), wybierze poprawną możliwość przez sprawdzenie czy znajduje się tam dane słowo. Oczywiście taki algorytm nie jest jednoznaczny, jednak powinien działać rozsądnie przy krótkich wiadomościach. Wiadomość oryginalna, szyfrowana oraz baza danych słów-kluczy maja być wczytywane z pliku.

10. Histogram z zadana podziałką

Napisz program, który 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

Napisz 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 i jeśli są – generować plik z wynikiem.

12. Gra w węża

Napisz program, który będzie 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 lub klawiatury. Wąż oczywiście rośnie po zdobyciu pokarmu oraz umiera po zderzeniu sam ze sobą albo ze ścianą.

13. Gra w życie

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

14. Ewakuacja płonącego budynku

Napisz 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 Instrukcji 5), przez co przez chwilę poruszają 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 zniknie, gdy do niego dotrze.

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

Napisz program symulujący maszynę do losowania używaną w Totolotku. Program w dużej mierze opiera się na Laboratorium 5 (zderzenia ze ścianami, kolizje), lecz należy uwzględnić także 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 inwencje użytkownika). Dodatkowo należy dodać niewielką ilość dyssypacji energii do programu, np. przy każdym zderzeniu niech piłki zachowują tylko 95 energii kinetycznej. Wreszcie, konieczna jest współpraca z myszką/klawiaturą: pierwsze kliknięcie zwalnia blokadę (zaczyna się symulacja), drugie rozpoczyna dmuchanie a trzecie zasysa piłki do boksu. Należy pamiętać, ze w boksie może być tylko jedna piłka. Oczywiście, piłki powinny się ze sobą zderzać. Należy w miarę możliwości wprowadzić lekkie zaburzenia na początku działania programu (losowo rozmieścić piłki w boksie), by wynik był niedeterministyczny.

Sterowanie przebiegiem programu za pomocą myszki i klawiatury

Mysz

- `mousedown()` – funkcja sprawdza czy naciśnięto klawisz myszki (zwraca 1 jeśli naciśnięto i 0 w przeciwnym wypadku)
- `mouseup()` – funkcja sprawdza czy po naciśnięciu zwolniono klawisz myszki (zwracana wartość – jw.)
- `whichmousebutton()` – funkcja zwraca 1 jeśli naciśnięto prawy klawisz i 0 jeśli naciśnięto lewy klawisz
- `mouseclickx()` – funkcja zwraca wartość współrzędnej x w momencie kliknięcia
- `mouseclicky()` – funkcja zwraca wartość współrzędnej y w momencie kliknięcia

Klawiatura

- `kbhit()` – funkcja zwraca 1 jeśli naciśnięto klawisz klawiatury i 0 w przeciwnym wypadku
- `getch()` – funkcja zwraca kod *ASCI* klikniętego klawisza