C++ obiektowy - Gra w statki

Ogólne

Program umożliwia utworzenie statku o określonych wymiarach na osi XY. Zadaniem użytkownika jest wskazać punkt, który pokrywa się ze statkiem. W wersji rozbudowanej, statek generowany jest automatycznie i pseudolosowo przez program (nie znamy jego położenia ani wielkości).

Uwaga: Nigdy nie wolno kopiować kodu z PDF-a, gdyż zawiera on niewidoczne znaki końca linii, twarde odstępy, odmienne cudzysłowy, odmienne apostrofy, odmienne myślniki. To godziny dodatkowej pracy na wykrywanie błędów i ich poprawianie. Kod należy przepisać ze zrozumieniem.

Kod - wersja podstawowa

Plik main.cpp

```
#include <iostream>
#include "statki.h"
using namespace std;

// Funkcja sprawdzająca, czy bombonierka dotarła na okręt. Jako argumenty wysyłamy całe
// obiekty, a nie ich liczne atrybuty. Nazwy argumentów mogą tutaj być dowolne (nie muszą
// odpowiadaćnazwom utworzonych obiektów).
void czyTrafiony(Cel bomb, Statek okr) {
   // Cztery warunki do sprawdzenia:
   if ((bomb.wspolrzednaX >= okr.naroznikX) && (bomb.wspolrzednaX <= (okr.naroznikX +
   okr.szerokoscStatku)) && (bomb.wspolrzednaY >= okr.naroznikY) && (bomb.wspolrzednaY
   <= (okr.naroznikY + okr.dlugoscStatku))) {
      cout << "\nBrawo! Udało ci się wysłać bombonierkę \"" << bomb.nazwaCelu << "\" na
      okręt \"" << okr.nazwaStatku << "\", który jest przyjacielem całego świata!";
   }
   else {
      cout << "\nNiestety, twoja bombonierka \"" << bomb.nazwaCelu << "\" nie dotarła do \""
      << okr.nazwaStatku << "\". Smuteczek!" << endl;
   }
}

int main() {
   cout << "===== GRA W STATKI=====\n" << endl;
   cout << "O nie! Kolejny lotniskowiec US zmierza w kierunku kolejnego państwa, aby
   zaprowadzić tam swoją amerykańską demokrację...\n" << endl;
   cout << "Jak zareagujesz? Co zrobisz?" << endl;
   cout << "\nPrzywitajmy go wysyłając mu bombonierkę z czekoladkami.\n" << endl;
   // Tworzymy obiekt klasy Cel:
   Cel bombonierka1("Merci!", 3, 3);
```

```
    // Tworzymy obiekt klasy Statek:
    Statek okret1("US Stealth Navy", 1, 1, 4, 5);

    cout << "Podaj nazwę bombonierki oraz współrzędne, do których ma zostać wystrzelona.
    Program odpowie, czy Twój prezent trafił do adresata." << endl;

    czyTrafiony(bombonierka1, okret1);

    cout << "\nNaciśnij ENTER, aby zakończyć..." << endl;
    cin.get();
    return 0;
}
```

Plik statki.h

```
#include <iostream>
using namespace std;

// Tylko zadeklarowanie klasy Statek, aby była widoczna w klasie Cel:
class Statek;

class Cel {
    private:
        string nazwaCelu;
        float wspolrzednaX, wspolrzednaY;
    public:
        // Definiujemy konstruktor:
        Cel(string nazwaCelu="A", float wspolrzednaX=0, float wspolrzednaY=0);
        // Tworzymy metodę pobierającą od użytkownika współrzędne celu:
        void wczytajCel();
        // Definiujemy przyjaźń tej klasy z zewnętrzną funkcją "czyTrafiony()",
        // dzięki czemu funkcja będzie miała dostęp do prywatnych atrybutów tej klasy.
        friend void czyTrafiony(Cel bomb, Statek okr);
};

class Statek {
    private:
        string nazwaStatku;
        // Aby utworzyć statek wystarczy znać współrzędne jednego narożnika
        // oraz długość i szerokość statku.
        float naroznikX, naroznikY, szerokoscStatku, dlugoscStatku;
    public:
        // Definiujemy konstruktor:
        Statek(string nazwaStatku="brak", float naroznikX=0, float naroznikY=0, float
        szerokoscStatku=1, float dlugoscStatku=1);
        // Inicjujemy metodę pobierającą od użytkownika położenie i wymiary statku:
        void wczytajStatek();
        // Definiujemy przyjaźń tej klasy z zewnętrzną funkcją "czyTrafiony()",
```

```
        // dzięki czemu funkcja będzie miała dostęp do prywatnych atrybutów tej klasy.
        friend void czyTrafiony(Cel bomb, Statek okr);
};
```

Plik statki.cpp

```
#include <iostream>
#include "statki.h"
using namespace std;

// Wywołujemy konstruktor. Wartości, które zostały odebrane przez parametry konstruktora,
// zostaną przypisane do prywatnych atrybutów klasy Cel:
Cel::Cel(string nazCelu, float x, float y) {
    nazwaCelu = nazCelu;
    wspolrzednaX = x;
    wspolrzednaY = y;
}

// Definiujemy metodę wczytajCel() klasy Cel:
void Cel::wczytajCel() {
    cout << "Podaj nazwę wysyłanej bombonierki: ";
    cin >> nazwaCelu;
    cout << "Podaj współrzędną X dla wysyłanej bombonierki: ";
    cin >> wspolrzednaX;
    cout << "Podaj współrzędną Y: ";
    cin >> wspolrzednaY;
}

// Wywołujemy konstruktor. Wartości, które zostały odebrane przez parametry konstruktora,
// zostaną przypisane do prywatnych atrybutów klasy Cel:
Statek::Statek(string nazStat, float x, float y, float szer, float dlug) {
    nazwaStatku = nazStat;
    naroznikX = x;
    naroznikY = y;
    szerokoscStatku = szer;
    dlugoscStatku = dlug;
}

// Definiujemy metodę wczytajStatek() klasy Statek:
void Statek::wczytajStatek() {
    cout << "Podaj nazwę okrętu: ";
    cin >> nazwaStatku;
    cout << "Podaj współrzędną X lewego narożnika: ";
    cin >> naroznikX;
    cout << "Podaj współrzędną Y lewego narożnika: ";
    cin >> naroznikY;
    cout << "Podaj szerokość okrętu: ";
    cin >> szerokoscStatku;
```

```
    cout << "Podaj długość okrętu: ";
    cin >> dlugoscStatku;
}
```

Kod - wersja rozbudowana (*Windows*)

Plik main.cpp

```
#include <iostream>
#include <windows.h> // Dla Sleep() w Windows, oraz dla PlaySound()
#include <cstdlib> // Dla srand()
#include <ctime> // Dla time()
#include "statki.h"
using namespace std;

// Funkcja sprawdzająca, czy bombonierka dotarła na okręt. Jako atrybuty wysyłamy całe
// obiekty, a nie ich liczne atrybuty. Nazwy obiektów mogą tutaj być dowolne (nie muszą
// odpowiadać nazwom utworzonych obiektów).
// Pamiętaj, że "bomb" i "okr" - to są kopie obiektów (dodatkowa pamięć). Jeśli chcemy
// pracować na oryginałach, musimy utworzyć referencje do obiektów.
void czyTrafiony(Cel &bomb, Statek &okr) {
    cout << "Naciśnij ENTER, aby wystrzelić bombonierkę..." << endl;
    cin.ignore();
    cin.get();
    // Cztery warunki do sprawdzenia:
    if ((bomb.wspolrzednaX >= okr.naroznikX) && (bomb.wspolrzednaX <= (okr.naroznikX +
    okr.szerokoscStatku)) && (bomb.wspolrzednaY >= okr.naroznikY) && (bomb.wspolrzednaY
    <= (okr.naroznikY + okr.dlugoscStatku))) {
        PlaySound(TEXT("explosion.wav"), NULL, SND_FILENAME | SND_ASYNC);
        Sleep(1000);
        cout << "\n\033[1;31;40mB U U U U M M M !!!\033[0m" << endl;
        Sleep(4000);
        cout << "\n\033[1;32;40mBrawo! Udało ci się wysłać bombonierkę \"" <<
        bomb.nazwaCelu << "\" na okręt \"" << okr.nazwaStatku << "\",\nktóry jest przyjacielem
        całego świata!\033[0m ;-)" << endl;
    }
    else {
        PlaySound(TEXT("missile.wav"), NULL, SND_FILENAME | SND_ASYNC);
        Sleep(1000);
        cout << "\n\033[1;31;40mNiestety, twoja bombonierka \"" << bomb.nazwaCelu << "\"
        nie dotarła do \"" << okr.nazwaStatku << "\".\nSmuteczek! :-(\033[0m" << endl;
    }
}

int main() {
    // ================= KOLOROWANIE NAPISÓW ===============
    // Tworzymy tzw. uchwyt do tego, co będzie pojawiać się na konsoli (do bufora konsoli):
    HANDLE konsola = GetStdHandle(STD_OUTPUT_HANDLE);
```

```
// Aktywujemy virtualny terminal i to, co będzie się na nim pojawiać:
#ifndef ENABLE_VIRTUAL_TERMINAL_PROCESSING // Jeśli nie jest zdefiniowany, to:
#define ENABLE_VIRTUAL_TERMINAL_PROCESSING 0x0004
#endif
// Aby powyższe działało, musimy aktywować i to, co poniżej:
#ifndef ENABLE_PROCESSED_OUTPUT // Jeśli nie jest zdefiniowany, to:
#define ENABLE_PROCESSED_OUTPUT 0x0001
#endif

// Wartość trybu (intup lub output). Słowo "dw" to skrót od "Display Window",
// jest to jednak nazwa zmiennej, i może być inna:
DWORD dwMode = 0;
dwMode |= ENABLE_PROCESSED_OUTPUT | ENABLE_VIRTUAL_TERMINAL_PROCESSING;
SetConsoleMode(konsola, dwMode);
// ============== KOLOROWANIE NAPISÓW - KONIEC =============

cout << "\n\033[1;34;40m===== GRA W STATKI=====\033[0m\n" << endl;
cout << "\033[1;30;40mO nie! Kolejny lotniskowiec US zmierza w kierunku kolejnego
państwa,\naby zaprowadzić tam swoją amerykańską demokrację...\n" << endl;
cout << "Jak zareagujesz? Co zrobisz?" << endl;
cout << "\nPrzywitajmy go wysyłając mu bombonierkę z czekoladkami.\n\n
==================== \033[0m\n" << endl;

cout << "Podaj nazwę bombonierki oraz współrzędne, do których ma zostać wystrzelona.\n
Program odpowie, czy Twój prezent trafił do adresata.\n" << endl;

// Tworzymy obiekt klasy Cel.
Cel bombonierka1;
// Obiekt z gotowymi wartościami:
// Cel bombonierka1("Merci!", 7, 3);

// Możemy też poprosić użytkownika o podanie wartości:
bombonierka1.wczytajCel();

// Tworzymy obiekt klasy Statek z gotowymi wartościami:
// Statek okret1("US Stealth Navy", 1, 1, 4, 5);
// Możemy też poprosić użytkownika o podanie wartości:
// okret1.wczytajStatek();
// Możemy także utworzyć pseudolosowe wartości położenia i wielkości statku:
srand(time(NULL)); // Inicjalizacja ziarna na podstawie bieżącego czasu
const int dolnyZakresXY = 0;
const int gornyZakresXY = 10;
const int dolnyZakresSzerDlug = 1;
const int gornyZakresSzerDlug = 6;
int wylosowanaLiczbaX = dolnyZakresXY + rand() % (gornyZakresXY - dolnyZakresXY + 1);
int wylosowanaLiczbaY = dolnyZakresXY + rand() % (gornyZakresXY - dolnyZakresXY + 1);
int wylosowanaLiczbaSzer = dolnyZakresSzerDlug + rand() % (gornyZakresSzerDlug -
dolnyZakresSzerDlug + 1);
```

```
    int wylosowanaLiczbaDlug = dolnyZakresSzerDlug + rand() % (gornyZakresSzerDlug -
    dolnyZakresSzerDlug + 1);
    Statek okret1("US Stealth Navy", wylosowanaLiczbaX, wylosowanaLiczbaY,
    wylosowanaLiczbaSzer, wylosowanaLiczbaDlug);

    czyTrafiony(bombonierka1, okret1);

    cout << "\n\n\033[1;30;40mPołożenie okrętu i jego wymiary:\nWspółrzędna X: " <<
    wylosowanaLiczbaX << ";\nWspółrzędna Y: " << wylosowanaLiczbaY << ";\nJego wymiary: "
    << wylosowanaLiczbaSzer << "x" << wylosowanaLiczbaDlug << "." << endl;
    cout << "\nNaciśnij ENTER, aby zakończyć...\033[0m" << endl;
    cin.get();
    return 0;
}
```

Plik statki.h

```
#include <iostream>

using namespace std;

// Tylko zadeklarowanie klasy Statek, aby była widoczna w klasie Cel:
class Statek;

class Cel {
    private:
        string nazwaCelu;
        float wspolrzednaX, wspolrzednaY;

    public:
        // Definiujemy konstruktor:
        Cel(string nazwaCelu="Merci!", float wspolrzednaX=0, float wspolrzednaY=0);
        // Deklarujemy metodę pobierającą od użytkownika współrzędne celu:
        void wczytajCel();
        // Definiujemy przyjaźń tej klasy z zewnętrzną funkcją "czyTrafiony()",
        // dzięki czemu funkcja będzie miała dostęp do prywatnych atrybutów tej klasy.
        friend void czyTrafiony(Cel &bomb, Statek &okr);
};


class Statek {
    private:
        string nazwaStatku;
        // Aby utworzyć statek wystarczy znać współrzędne jednego narożnika
        // oraz długość i szerokość statku.
        float naroznikX, naroznikY, szerokoscStatku, dlugoscStatku;
    public:
        // Definiujemy konstruktor:
```

```
    Statek(string nazwaStatku="US Stealth Navy", float naroznikX=0, float naroznikY=0,
    float szerokoscStatku=1, float dlugoscStatku=1);
    // Tworzymy metodę pobierającą od użytkownika położenie i wymiary statku:
    void wczytajStatek();
    // Definiujemy przyjaźń tej klasy z zewnętrzną funkcją "czyTrafiony()",
    // dzięki czemu funkcja będzie miała dostęp do prywatnych atrybutów tej klasy.
    friend void czyTrafiony(Cel &bomb, Statek &okr);
};
```

Plik statki.cpp

```
#include <iostream>
#include <limits> // Dla numeric_limits
#include <Windows.h> // Dla PlaySound()
#include "statki.h"

using namespace std;

// Wywołujemy konstruktor. Wartości, które zostały odebrane przez parametry konstruktora,
// zostaną przypisane do prywatnych atrybutów klasy Cel:
Cel::Cel(string nazCelu, float x, float y) {
   nazwaCelu = nazCelu;
   wspolrzednaX = x;
   wspolrzednaY = y;
}

// Definiujemy metodę wczytajCel() klasy Cel:
void Cel::wczytajCel() {
   cout << "Podaj nazwę wysyłanej bombonierki (np. Merci!): ";
   getline(cin, nazwaCelu);
   if (nazwaCelu == "") {
      nazwaCelu = "Merci!";
   }
   cout << "Podaj współrzędną X dla wysyłanej bombonierki (0-16): ";
   // Tylko wprowadzenie liczby daje prawdę:
   while(!(cin >> wspolrzednaX)) {
      cout << "To nie jest liczba całkowita. Spróbuj ponownie: ";
      cin.clear(); // Wyczyść stan błędu
      // Wyczyść bufor, aby uniknąć nieskończonej pętli, gdy użytkownik wprowadzi
      // nieprawidłowe dane.
      // numeric_limits - określa maksymalną wartość typu danych <streamsize> lub innego
      // typu danych (maksymalną pojemność);
      // <streamsize>::max() - wielkość bufora ustawiony na maksymalny; oznacza ignorowanie
      // maksymalnej ilości danych w buforze;
      // '\n' - znak, który chcemy zignorować;
      cin.ignore(numeric_limits <streamsize>::max(), '\n');
   }
   PlaySound(TEXT("pumping-shotgun.wav"), NULL, SND_FILENAME | SND_ASYNC);
```

```
    cout << "Podaj współrzędną Y (0-16): ";
    while(!(cin >> wspolrzednaY)) {
        cout << "To nie jest liczba całkowita. Spróbuj ponownie: ";
        cin.clear(); // Wyczyść stan błędu
        cin.ignore(numeric_limits < streamsize>::max(), '\n'); // Wyczyść bufor
    }
    PlaySound(TEXT("pumping-shotgun.wav"), NULL, SND_FILENAME | SND_ASYNC);
}


// Wywołujemy konstruktor. Wartości, które zostały odebrane przez parametry konstruktora,
// zostaną przypisane do prywatnych atrybutów klasy Cel:
Statek::Statek(string nazStat, float x, float y, float szer, float dlug) {
    nazwaStatku = nazStat;
    naroznikX = x;
    naroznikY = y;
    szerokoscStatku = szer;
    dlugoscStatku = dlug;
}

// Definiujemy metodę wczytajStatek() klasy Statek:
void Statek::wczytajStatek() {
    cout << "Podaj nazwę okrętu: ";
    getline(cin, nazwaStatku);
    if (nazwaStatku == "") {
        nazwaStatku = "US Stealth Navy";
    }
    cout << "Podaj współrzędną X lewego narożnika (0-20): ";
    while(!(cin >> naroznikX)) {
        cout << "To nie jest liczba całkowita. Spróbuj ponownie: ";
        cin.clear(); // Wyczyść stan błędu
        cin.ignore(numeric_limits < streamsize>::max(), '\n'); // Wyczyść bufor
    }
    cout << "Podaj współrzędną Y lewego narożnika (0-20): ";
    while(!(cin >> naroznikY)) {
        cout << "To nie jest liczba całkowita. Spróbuj ponownie: ";
        cin.clear(); // Wyczyść stan błędu
        cin.ignore(numeric_limits < streamsize>::max(), '\n'); // Wyczyść bufor
    }
    cout << "Podaj szerokość okrętu (1-6): ";
    while(!(cin >> szerokoscStatku)) {
        cout << "To nie jest liczba całkowita. Spróbuj ponownie: ";
        cin.clear(); // Wyczyść stan błędu
        cin.ignore(numeric_limits < streamsize>::max(), '\n'); // Wyczyść bufor
    }
    cout << "Podaj długość okrętu (1-6): ";
    while(!(cin >> dlugoscStatku)) {
        cout << "To nie jest liczba całkowita. Spróbuj ponownie: ";
```

```
        cin.clear(); // Wyczyść stan błędu
        cin.ignore(numeric_limits < streamsize>::max(), '\n'); // Wyczyść bufor
    }
}
// Do działania funkcji PlaySound() potrzebne będzie podlinkowanie biblioteki libwinmm.a, czyli:
// Settings / Compiler / Linker Settings / Link Libraries:
// C:\Program Files\CodeBlocks\MinGW\x86_64-w64-mingw32\lib\libwinmm.a
```

Kod - wersja rozszerzona (Linux)

Plik main.cpp

```
#include <iostream>
#include <unistd.h> // Dla sleep() w Linux
// #include <windows.h> // Dla Sleep() w Windows
#include "statki.h"
using namespace std;

// Funkcja sprawdzająca, czy bombonierka dotarła na okręt.
// Jako atrybuty wysyłamy całe obiekty, a nie ich liczne atrybuty.
// Nazwy obiektów mogą tutaj być dowolne (nie muszą odpowiadać
// nazwom utworzonych obiektów).
// Pamiętaj, że "bomb" i "okr" - to są kopie obiektów (dodatkowa pamięć).
// Jeśli chcemy pracować na orginałach, musimy utworzyć referencje do obiektów.
void czyTrafiony(Cel &bomb, Statek &okr) {
    cout << "Naciśnij ENTER, aby wystrzelić bombonierkę..." << endl;
    cin.ignore();
    cin.get();
    // Cztery warunki do sprawdzenia:
    if ((bomb.wspolrzednaX >= okr.naroznikX) && (bomb.wspolrzednaX <= (okr.naroznikX +
    okr.szerokoscStatku)) && (bomb.wspolrzednaY >= okr.naroznikY) && (bomb.wspolrzednaY
    <= (okr.naroznikY + okr.dlugoscStatku))) {
        system("play -q explosion.wav &");
        sleep(1); // Dla systemu Windows: Sleep(1000);
        cout << "\n\033[1;31;40mB U U U U M M M !!!\033[0m" << endl;
        sleep(4);
        cout << "\n\033[1;32;40mBrawo! Udało ci się wysłać bombonierkę \"" <<
        bomb.nazwaCelu << "\" na okręt \"" << okr.nazwaStatku << "\",\nktóry jest przyjacielem
        całego świata!\033[0m ;-)" << endl;
    }
    else {
        system("play -q missile.wav &");
        sleep(1);
        cout << "\n\033[1;31;40mNiestety, twoja bombonierka \"" << bomb.nazwaCelu << "\"
        nie dotarła do \"" << okr.nazwaStatku << "\".\nSmuteczek! :-(\033[0m" << endl;
    }
}
```

```
int main() {
    cout << "\n\033[1;34;40m===== GRA W STATKI =====\033[0m\n" << endl;
    cout << "\033[1;30;40mO nie! Kolejny lotniskowiec US zmierza w kierunku kolejnego
    państwa, \naby zaprowadzić tam swoją amerykańską demokrację...\n" << endl;
    cout << "Jak zareagujesz? Co zrobisz?" << endl;
    cout << "\nPrzywitajmy go wysyłając mu bombonierkę z czekoladkami.\n\n
    ========================\033[0m\n" << endl;

    cout << "Podaj nazwę bombonierki oraz współrzędne, do których ma zostać wystrzelona.\n
    Program odpowie, czy Twój prezent trafił do adresata.\n" << endl;

    // Tworzymy obiekt klasy Cel:
    Cel bombonierka1;
    // Obiekt z gotowymi wartościami:
    // Cel bombonierka1("Merci!", 7, 3);

    // Możemy też poprosić użytkownika o podanie wartości:
    bombonierka1.wczytajCel();

    // Tworzymy obiekt klasy Statek z gotowymi wartościami:
    // Statek okret1("US Stealth Navy", 1, 1, 4, 5);
    // Możemy też poprosić użytkownika o podanie wartości:
    // okret1.wczytajStatek();
    // Możemy także utworzyć pseudolosowe wartości położenia i wielkości statku:
    srand(time(NULL)); // Inicjalizacja ziarna na podstawie bieżącego czasu
    const int dolnyZakresXY = 0;
    const int gornyZakresXY = 20;
    const int dolnyZakresSzerDlug = 1;
    const int gornyZakresSzerDlug = 6;
    int wylosowanaLiczbaX = dolnyZakresXY + rand() % (gornyZakresXY - dolnyZakresXY + 1);
    int wylosowanaLiczbaY = dolnyZakresXY + rand() % (gornyZakresXY - dolnyZakresXY + 1);
    int wylosowanaLiczbaSzer = dolnyZakresSzerDlug + rand() % (gornyZakresSzerDlug -
    dolnyZakresSzerDlug + 1);
    int wylosowanaLiczbaDlug = dolnyZakresSzerDlug + rand() % (gornyZakresSzerDlug -
    dolnyZakresSzerDlug + 1);
    Statek okret1("US Stealth Navy", wylosowanaLiczbaX, wylosowanaLiczbaY,
    wylosowanaLiczbaSzer, wylosowanaLiczbaDlug);

    czyTrafiony(bombonierka1, okret1);

    cout << "\n\n\033[1;30;40mPołożenie okrętu i jego wymiary:\nWspółrzędna X: " <<
    wylosowanaLiczbaX << ";\nWspółrzędna Y: " << wylosowanaLiczbaY << ";\nJego wymiary: "
    << wylosowanaLiczbaSzer << "x" << wylosowanaLiczbaDlug << "." << endl;
    cout << "\nNaciśnij ENTER, aby zakończyć...\033[0m" << endl;
    cin.get();
    return 0;
}
```

Plik statki.h

```
#include <iostream>
using namespace std;

// Tylko zadeklarowanie klasy Statek, aby była widoczna w klasie Cel:
class Statek;

class Cel {
   private:
      string nazwaCelu;
      float wspolrzednaX, wspolrzednaY;

   public:
      // Definiujemy konstruktor:
      Cel(string nazwaCelu="Merci!", float wspolrzednaX=0, float wspolrzednaY=0);
      // Deklarujemy metodę pobierającą od użytkownika współrzędne celu:
      void wczytajCel();
      // Definiujemy przyjaźń tej klasy z zewnętrzną funkcją "czyTrafiony()",
      // dzięki czemu funkcja będzie miała dostęp do prywatnych atrybutów tej klasy.
      friend void czyTrafiony(Cel &bomb, Statek &okr);
};


class Statek {
   private:
      string nazwaStatku;
      // Aby utworzyć statek wystarczy znać współrzędne jednego narożnika
      // oraz długość i szerokość statku.
      float naroznikX, naroznikY, szerokoscStatku, dlugoscStatku;
   public:
      // Definiujemy konstruktor:
      Statek(string nazwaStatku="US Stealth Navy", float naroznikX=0, float naroznikY=0,
      float szerokoscStatku=1, float dlugoscStatku=1);
      // Tworzymy metodę pobierającą od użytkownika położenie i wymiary statku:
      void wczytajStatek();
      // Definiujemy przyjaźń tej klasy z zewnętrzną funkcją "czyTrafiony()",
      // dzięki czemu funkcja będzie miała dostęp do prywatnych atrybutów tej klasy.
      friend void czyTrafiony(Cel &bomb, Statek &okr);
};
```

Plik statki.cpp

```
#include <iostream>
#include <limits> // Dla numeric_limits
#include "statki.h"

using namespace std;
```

```
// Wywołujemy konstruktor. Wartości, które zostały odebrane przez parametry konstruktora,
// zostaną przypisane do prywatnych atrybutów klasy Cel:
Cel::Cel(string nazCelu, float x, float y) {
    nazwaCelu = nazCelu;
    wspolrzednaX = x;
    wspolrzednaY = y;
}

// Definiujemy metodę wczytajCel() klasy Cel:
void Cel::wczytajCel() {
    cout << "Podaj nazwę wysyłanej bombonierki (np. Merci!): ";
    getline(cin, nazwaCelu);
    if (nazwaCelu == "") {
        nazwaCelu = "Merci!";
    }
    cout << "Podaj współrzędną X dla wysyłanej bombonierki (0-26): ";
    // Tylko wprowadzenie liczby daje prawdę:
    while(!(cin >> wspolrzednaX)) {
        cout << "To nie jest liczba całkowita. Spróbuj ponownie: ";
        cin.clear(); // Wyczyść stan błędu
        // Wyczyść bufor, aby uniknąć nieskończonej pętli, gdy użytkownik wprowadzi
        // nieprawidłowe dane.
        // numeric_limits - określa maksymalną wartość typu danych <streamsize> lub innego
        // typu danych (maksymalną pojemność);
        // <streamsize>::max() - wielkość bufora ustawiony na maksymalny; oznacza ignorowanie
        // maksymalnej ilości danych w buforze;
        // '\n' - znak, który chcemy zignorować;
        cin.ignore(numeric_limits <streamsize>::max(), '\n');
    }
    system("play -q pumping-shotgun.wav &");
    cout << "Podaj współrzędną Y (0-26): ";
    while(!(cin >> wspolrzednaY)) {
        cout << "To nie jest liczba całkowita. Spróbuj ponownie: ";
        cin.clear(); // Wyczyść stan błędu
        cin.ignore(numeric_limits < streamsize>::max(), '\n'); // Wyczyść bufor
    }
    system("play -q pumping-shotgun.wav &");
}


// Wywołujemy konstruktor. Wartości, które zostały odebrane przez parametry konstruktora,
// zostaną przypisane do prywatnych atrybutów klasy Cel:
Statek::Statek(string nazStat, float x, float y, float szer, float dlug) {
    nazwaStatku = nazStat;
    naroznikX = x;
    naroznikY = y;
    szerokoscStatku = szer;
    dlugoscStatku = dlug;
```

```
}

// Definiujemy metodę wczytajStatek() klasy Statek:
void Statek::wczytajStatek() {
    cout << "Podaj nazwę okrętu: ";
    getline(cin, nazwaStatku);
    if (nazwaStatku == "") {
        nazwaStatku = "US Stealth Navy";
    }
    cout << "Podaj współrzędną X lewego narożnika (0-20): ";
    while(!(cin >> naroznikX)) {
        cout << "To nie jest liczba całkowita. Spróbuj ponownie: ";
        cin.clear(); // Wyczyść stan błędu
        cin.ignore(numeric_limits < streamsize>::max(), '\n'); // Wyczyść bufor
    }
    cout << "Podaj współrzędną Y lewego narożnika (0-20): ";
    while(!(cin >> naroznikY)) {
        cout << "To nie jest liczba całkowita. Spróbuj ponownie: ";
        cin.clear(); // Wyczyść stan błędu
        cin.ignore(numeric_limits < streamsize>::max(), '\n'); // Wyczyść bufor
    }
    cout << "Podaj szerokość okrętu (1-6): ";
    while(!(cin >> szerokoscStatku)) {
        cout << "To nie jest liczba całkowita. Spróbuj ponownie: ";
        cin.clear(); // Wyczyść stan błędu
        cin.ignore(numeric_limits < streamsize>::max(), '\n'); // Wyczyść bufor
    }
    cout << "Podaj długość okrętu (1-6): ";
    while(!(cin >> dlugoscStatku)) {
        cout << "To nie jest liczba całkowita. Spróbuj ponownie: ";
        cin.clear(); // Wyczyść stan błędu
        cin.ignore(numeric_limits < streamsize>::max(), '\n'); // Wyczyść bufor
    }
}
```

Ostatnia aktualizacja: 26 kwietnia 2024.