Zestaw zadań C++, część II

20 stycznia 2019

Zadanie 1 Powtórka z poprzednich zajęć: napisz program, który zamieni wartości dwóch zmiennych, tzn. dla zmiennych a i b przypisze do zmiennej a wartość b, a do b wartość zmiennej a. Wypisać na ekran wartości przed i po zamianie.

Zadanie 2 Do ostatniego zadania z poprzedniego zestawu (wyznaczanie pierwiastków równania kwadratowego) dodać sprawdzenie, czy Δ jest nieujemna. W przypadku gdy $\Delta < 0$ wypisać odpowiedni komunikat. Spróbować dodać warunek $\Delta == 0$, aby wypisać podwójny pierwiastek x_0 zamiast x_1 i x_2.

Zadanie 3 Napisać program, który pyta użytkownika o podanie liczby innej niż 44 tak długo, aż użytkownik nie poda liczby 44. Gdy użytkownik poda 44 wypisać na ekran odpowiedni komunikat i zakończyć działanie programu. *Zmodyfikować program tak, aby zakończyć pętlę także gdy liczba prób była większa od 10.

Zadanie 4 Po wykonaniu każdego z poniższych podpunktów wypisać na ekran zawartość tablicy.

- Utworzyć tablicę o rozmiarze N,
- wyzerować wszystkie elementy tablicy,
- wypełnić tablicę kwadratami kolejnych liczb naturalnych,
- wyznaczyć sumę oraz średnią wszystkich elementów tablicy.

Zadanie 5

- Utworzyć strukturę Circle posiadającą pola:
  - *color* type string
  - *x, y, radius* typu double
- Utworzyć 3 zmienne typu Circle, przypisać polom dowolne wartości
- Poprosić użytkownika o podanie współrzędnych punktu: x oraz y
- Sprawdzić, czy podany punkt leży wewnątrz jednego z okręgów, jeżeli tak, wypisać na ekran kolor okręgu oraz jego pole

Zadanie 6 Przygotować strukturę Person z polami *name* oraz *height*. Wykorzystać klasę vector do przechowania 5 zmiennych typu Person, wstawić dowolne imiona. Napisać program który zapyta użytkownika o wzrost tych 5 osób (wykorzystać pętlę oraz cin). Następnie:

- Wypisać która osoba jest najniższa

1 Rozwiązania

Zadanie 1
```cpp
#include <iostream>
int main()
{
	int a = 2, b = 3;
	std::cout<<"Before swap: "<<a<<" "<<b<<"\n";

	int temp;
	temp = a;
	a = b;
	b = temp;

	std::cout<<"After swap: "<<a<<" "<<b<<"\n";
	return 0;
}
```

Zadanie 2
```cpp
#include <iostream>
#include <cmath>

int main()
{
	double a = 1, b = -1, c = -1;
	double delta = b*b - 4*a*c;
	if (delta < 0)
	{
		std::cout<<"No roots\n";
	}
	else
	{
		double deltaSqrt = std::sqrt(delta);
		double x1 = (-b - deltaSqrt)/(2*a);
		double x2 = (-b + deltaSqrt)/(2*a);

		std::cout<<a<<"x^2 + "<<b<<"x + "<<c;
		std::cout<<" roots (x1, x2): ("<<x1<<","<<x2<<")\n";
	}

	return 0;
}
```

Zadanie 3
```cpp
#include <iostream>

int main()
{
	int count = 0;
	while(count < 11) //while (true) dla pierwszej wersji
	{
		std::cout<<"Enter number different than 44: ";
		int x;
		std::cin>>x;
```

```
                    if (x == 44)
                    {
                            break;
                    }
            }
            return 0;
}
```

Zadanie 4

```
#include <iostream>

int main()
{
        const int N = 10;
        int tab[N];

        double sum = 0;
        for (int i = 0; i < N; i++)
        {
                tab[i] = 0;
                tab[i] = i*i;
                sum += tab[i];
        }
        double avg = sum/N;

        return 0;
}
```

Zadanie 5

```
#include <iostream>
#include <string>
#include <cmath>

struct Circle
{
        std::string color;
        double x, y, radius;
};

int main()
{
        Circle c1, c2, c3;
        c1.color = "Red"; c2.color = "Blue"; c3.color = "Green";
        c1.x = 1; c1.y = 1; c1.radius = 10;
        c2.x = 3; c2.y = -1; c2.radius = 3;
        c3.x = 0; c3.y = 0; c3.radius = 1;

        double x, y;
        std::cout<<"Podaj x, y: ";
        std::cin>>x>>y;

        //dla jednego okregu
        double dx = c1.x - x, dy = c1.y - y;
```

```
        double dist = std::sqrt(dx*dx + dy*dy);
        if (dist <= c1.radius)
        {
                double area = M_PI*c1.radius*c1.radius;
                std::cout<<"Color: "<<c1.color
                        <<", area: "<<area<<"\n";
        }

        return 0;
}
```

Zadanie 6 Kompilować z flagą -std=c++11: g++ -std=c++11 main.cpp

```
#include <iostream>
#include <vector>

struct Person
{
        std::string name;
        int height;
};

int main()
{
        std::vector<Person> persons(5);
        persons[0].name = "Adam";
        persons[1].name = "Ewa";
        persons[2].name = "Pawel";
        persons[3].name = "Radek";
        persons[4].name = "Ania";

        for (Person &p : persons)
        {
                std::cout<<"Podaj wzrost "<<p.name<<"\n";
                std::cin>>p.height;
        }

        int minheight = persons[0].height;
        int minID = 0;

        for (int i = 1; i < persons.size(); i++)
        {
                if (minheight > persons[i].height)
                {
                        minheight = persons[i].height;
                        minID = i;
                }
        }
        std::cout<<"Najnizsza osoba: "<<persons[minID].name<<" "
                <<persons[minID].height<<"\n";

        return 0;
```

```
}
```

2 Projekt

Przygotować następujący program:

- Utworzyć strukturę *Particle* posiadającą pola rzeczywiste:
  - Położenie: x, y
  - Prędkość: vx, vy
  - Wysokość: h
  - Najniższa znana wysokośc: h_min
  - Najniższa znana pozycja: x_min, y_min
- Utworzyć tablice typu *Particle* o rozmiarze 25.
- Rozmieścić obiekty na przestrzeni $-5 <= x,\ y <= 5$ w dowolny sposób, można: tab[i].x = 2*(i/5) - 4, tab[i].y = 2*(i%5) - 4.
- Wyznaczyć wysokość dla każdego elementu. Wysokość h dana jest wzorem:

$$h(x, y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2 \tag{1}$$

  Dla każdego elemntu sprawdzić, czy jest to najniższa do tej pory odwiedzona przez ten element wysokość, jeżeli tak, zmienić odpowiednio pola x_min, y_min oraz h_min.
- Znaleźć x_g oraz y_g jako x i y najniżej położonej cząsteczki
- Dla każdego elementu wyznaczyć wartości vx oraz vy z następującego wzoru:

$$v_i = 0.9v_i + 2r_1 \times (i_g - i) + 2r_2 \times (i\_min - i) \tag{2}$$

  gdzie r_1 i r_2 to liczby losowe z przedziału $[0, 1]$ - losowane za każdym razem dla każdej cząsteczki, a i to x oraz y (2 równania). Wykorzystując bibliotekę cstdlib można wygenerować losową liczbę od 0 do 1 w bardzo uproszczony sposób:

$$(rand()\%1001)/1000.0 \tag{3}$$

- Uaktualnić położenia elementów zgodnie ze wzorem:

$$i = i + v_i \tag{4}$$

  , gdzie tak samo jak wcześniej i to x oraz y.
- Powtórzyć 4 poprzednie punkty 10000 razy (dodać zewnętrzną pętlę)
- Wyznaczyć najniżej położony element i wypisać jego współrzędzne oraz wysokość na ekran.