Podstawy programowania
Materiały dydaktyczne do laboratorium

26 września 2016

Zadania domowe

1 Przygotowanie środowiska i pierwsze programy

Zadanie 1. Zainstaluj na własnym komputerze kompilator `C++` zgodnie z zaleceniami prowadzącego zajęcia. Stwórz pierwszy projekt. Napisz, skompiluj i uruchom na swoim komputerze pierwszy program.

Przykład 1. Pusty program

```
int main() {
  return 0;
}
```

Zadanie 2*. Skompiluj i uruchom program jeszcze raz, tym razem używając kompilatora z linii poleceń[1].

Zadanie 3. Dołącz bibliotekę `iostream`, która pozwoli zobaczyć pierwszy efekt działania programu. Użyj strumienia `cout` do wyświetlenia pierwszego napisu.

Przykład 2. Witaj

```
#include<iostream>

int main() {
  std::cout << "Witaj!" << std::endl;
  return 0; /* zakończ program */
}
```

[1]Znak * sygnalizuje zadanie nieco trudniejsze, którego prawidłowe wykonanie nie jest niezbędne do wykonania kolejnych zadań - można je na razie pominąć i wrócić do niego później. Podobnie ** sygnalizuje zadania jeszcze trudniejsze, ich wykonanie może wymagać dodatkowej wiedzy lub chwili głębszego zastanowienia.

Zadanie 4. Odszukaj skompilowany program na dysku, uruchom go poza środowiskiem, korzystając z konsoli.

Zadanie 5. Uruchom swój program jeszcze raz, tym razem przekierowując strumień standardowego wyjścia (czyli to, co jest wypisywane jako wynik działania programu) do pliku. Zakładając, że program po skompilowaniu nazywa się `witaj.exe`, a wyniki chcemy mieć w pliku `wyniki.txt`, należy wydać komendę:

```
witaj > wyniki.txt
```

2 Typy i zmienne

Zadanie 6. Uruchom kolejny program, przekonaj się o konieczności nadawania zmiennym wartości początkowych. Sprawdź, czy wykorzystywane przez Ciebie środowisko ostrzega Cię przed tego typu błędami (Visual Studio ostrzega).

```
#include<iostream>

int main() {
  int x = 0;    // deklaracja i inicjalizacja zmiennej
  std::cout << x << std::endl;  // 0
  int y;        // tylko deklaracja zmiennej
  std::cout << y << std::endl;     // błąd! wyświetlony
                // wynik jest
                // przypadkową liczbą
  return 0;
}
```

Zadanie 7. Pozbądź się konieczności ciągłego używania `std` w kodzie programu, skorzystaj z deklaracji `using namespace std;`

Zadanie 8. Napisz program, który na podstawie wartości zmiennych całkowitych a i b obliczy obwód prostokąta o wymiarach a i b.

Zadanie 9. W programie z zadania 8 wczytaj wartości zmiennych ze standardowego wejścia używając strumienia `cin`.

3 Instrukcja wyboru

Jeśli w zależności od danych wejściowych lub stanu zmiennych przewidujemy różny przebieg sterowania (różny zestaw instrukcji), to stosujemy instrukcje wyboru.

Przykład 3. Mniejsza z dwóch liczb

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

int main() {
  int a;
  int b;
  cin >> a >> b;
```

```
  if (a<b)
    cout << a;
  else
    cout << b;
  cout << endl;
  return 0;
}
```

Zadanie 10. Przyjrzyj się programowi poniżej:

Przykład 4.

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

int main() {
  double x;
  double y;

  cout << "Ten program dodaje dwie liczby." << endl;
  cout << "Podaj liczbe 1 : ";
  if(cin >> x) {
    cout << "Podaj liczbe 2 : ";
    if (cin >> y) {
      cout << x << " + " << y << " = " << (x + y) << endl;
    }
    else {
      cout << "niepoprawne dane" << endl;
     }
  }
  else {
    cout << "niepoprawne dane" << endl;
  }
  cout << "koniec." << endl;
  return 0;
}
```

Wczytanie danych może się nie powieść, wtedy wyrażenie (cin » y) ma wartość 0.

Zadanie 11. Zastosuj sprawdzanie poprawności wczytanych danych do programu obliczającego obwód prostokąta (zadanie 8).

Zadanie 12. Dane są trzy liczby a, b i c. Napisz program, który znajduje najmniejszą (największą, środkową) z nich.

Zadanie 13. Dane są dwie pary liczb $\{a, b\}$ i $\{c, d\}$. Napisz program, który sprawdza, czy $\{a, b\} = \{c, d\}$.

Zadanie 14. Dane są trzy liczby a, b i c. Napisz program, który sprawdza, czy z boków o podanej długości można zbudować trójkąt.

4 Pętla `for`

Wykonanie grupy instrukcji określoną liczbę razy (tutaj 7 razy) przedstawia następujący program:

Przykład 5.

```
#include<iostream>
using namespace std;
int main() {
  for (int i = 0; i<7; ++i) {
    cout << "Witaj przyjacielu" << endl;
  }
  return 0;
}
```

Poniższy program wypisuje kolejne liczby od 0 do 9. Do tego celu wykorzystano licznik pętli. W poniższym przykładzie licznikiem jest zmienna `i`:

Przykład 6.

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

int main() {
  for (int i = 0; i < 10; i++) {
    cout << i << endl;
  }
  return 0;
}
```

Kolejny przykład wypisuje liczby z przedziału $[100, 200]$ ze skokiem 42:

Przykład 7.

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

int main() {
  for (int i = 100; i <= 200; i+=42) {
    cout << i << endl;
  }
  return 0;
}
```

Zadanie 15. Używając pętli napisz program, który wypisuje kolejne liczby od 4 do 17 włącznie.

Zadanie 16. Używając pętli napisz program, który wypisuje kolejne liczby od 14 do 7 włącznie (14, 13, ..., 7).

Zadanie 17. Dane są dwie liczby całkowite a i b, $a < b$. Napisz program, który wypisuje liczby parzyste z zakresu $[a, b]$.

Zadanie 18. Napisz program, który przepisuje kolejne dodatnie liczby parzyste podane przez użytkownika, nieparzyste pomija a kończy działanie po podaniu `0` lub liczby ujemnej. Wskazówka: użyj `break`.

Zadanie 19*. Przeanalizuj poniższy przykład, napisz program, który wyświetla zadaną liczbę w postaci sumy potęg liczby 2.

Przykład 8.

```
#include<iostream>
using namespace std;
int main()
{
  int x;
  cin >> x;
  int exp = 0;
  int rem = x;
  while (rem%2 == 0) {
    rem>>=1;
    exp++;
  }
  if (exp>0) {
    cout << x << " = " << "2^" << exp;
    if (rem>1) cout << "*" << rem;
  }
  else {
    cout << x << " to liczba nieparzysta" << endl;
  }
  return 0;
}
```

5 Pętle zagnieżdżone

Jeśli chcemy powtarzać wielokrotnie wykonanie tej samej pętli, to możemy umieścić jedną pętlę wewnątrz drugiej. Mówimy wtedy o *pętlach zagnieżdżonych*.

Zadanie 20. Przeanalizuj i porównaj dwa poniższe programy

Przykład 9. Prostokąt z gwiazdek, metoda 1

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

int main()
{
  int sizeX = 10;
  int sizeY = 20;

  for (int j=0; j<sizeX; ++j) {
    cout << '*';
  }
```

```
  cout << endl;
  for (int i=1;i<sizeY-1; ++i) {
    cout << '*';
    for (int j=1; j<sizeX-1; ++j) {
      cout << ' ';
    }
    cout << '*' << endl;
  }
  for (int j=0; j<sizeX; ++j) {
    cout << '*';
  }
  cout << endl;
  return 0;
}
```

Przykład 10. Prostokąt z gwiazdek, metoda 2

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

int main()
{
  int sizeX = 10;
  int sizeY = 20;

  for (int i=0;i<sizeY; ++i) {
    for (int j=0; j<sizeX; ++j) {
      if (i==0 || i==sizeY-1 || j==0 || j==sizeX-1) {
        cout << '*';
      } else {
        cout << ' ';
      }
    }
    cout << endl;
  }
  return 0;
}
```

W przykładzie 9 metodą prowadzącą do rozwiązania jest analiza pożądanego kształtu pod kątem powtarzających się elementów. W przypadku prostokąta są to linie w całości wypełnione gwiazdkami (pierwsza i ostatnia) oraz linie, w których gwiazdka występuje tylko na pierwszym i ostatnim miejscu.

W przykładzie 10 stosujemy nieco inne podejście. Drukujemy znak po znaku obliczając na podstawie współrzędnych (numeru wiersza i numeru znaku w wierszu) jaki znak należy wypisać.

Zadanie 21. Napisz dwa programy, które drukują z gwiazdek duży znak `X`. Podobnie jak w przykładach zastosuj obie metody prowadzące do rozwiązania. Porównaj otrzymane kody pod względem stopnia skomplikowania i możliwości wprowadzenia zmian.

Przykład 11. Duży `X` z gwiazdek, metoda 1

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

int main() {
  int size = 11;
  // gorne V
  for (int i=0;i<size/2; ++i) {
    for (int j=0; j<i; ++j) {
      cout << ' ';
    }
    cout << '*';
    for (int j=0; j<size-2*i-2; ++j) {
      cout << ' ';
    }
    cout << '*';
    for (int j=0; j<size; ++j) {
      cout << ' ';
    }
    cout << endl;
  }
  // srodkowy wiersz
  if (size%2==1) {
    for (int j=0; j<size/2; ++j) {
      cout << ' ';
    }
    cout << '*';
    for (int j=0; j<size/2; ++j) {
      cout << ' ';
    }
    cout << endl;
  }
  // odwrocone V
  for (int i=0;i<size/2; ++i) {
    for (int j=0; j<size/2-i-1; ++j) {
      cout << ' ';
    }
    cout << '*';
    for (int j=0; j<2*i+size%2; ++j) {
      cout << ' ';
    }
    cout << '*';
    for (int j=0; j<size/2-i-1; ++j) {
      cout << ' ';
    }
    cout << endl;
  }
  return 0;
```

```
}
```

Przykład 12. Duży `X` z gwiazdek, metoda 2

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

int main()
{
  int size = 13;
  for (int i=0;i<size; ++i) {
    for (int j=0; j<size; ++j) {
      if (i==j || i==size-j-1) {
        cout << '*';
      } else {
        cout << ' ';
      }
    }
    cout << endl;
  }
  return 0;
}
```

Praca na zajęciach

Zadanie 1. Napisz program, który drukuje wskazany kształt o zadanych rozmiarach. Przykładowe kształty poniżej.

```
./\.
/..\
\../
.\/.
./\.
/..\
\../
.\/.
./\.
/..\
\../
.\/.
```

```
**********
*\.*./*\.*./*\.*
*.\*/.*.\*/.*.\*
**********
*./*\.*./*\.*./*
*/.*.\*/.*.\*/.*
**********
```

```
**********
*........*
*........*
**********
*.....*\.*
*.....*.\*
**********
```