Politechnika Wrocławska

Techniki programowania

INP001002Wl

rok akademicki 2018/19

semestr letni

Wykład 1

Karol Tarnowski

karol.tarnowski@pwr.edu.pl

A-1 p. 411B

Politechnika Wrocławska

Plan wykładów

- Wprowadzenie do języka C++
- Programowanie obiektowe
- Wykorzystanie biblioteki
  - STL – Standard Template Library
  - GSL – GNU Scientific Library

Politechnika Wrocławska

Plan prezentacji (1)

- Pierwszy program w C++
- Interakcja z użytkownikiem
  - `cout`
  - `cin`
- Typ `string`
- Typ `bool`

Na podstawie:

- A. Allain, *Przewodnik dla początkujących C++*
- S. Prata, *Szkoła programowania C++*

Politechnika Wrocławska

Plan prezentacji (2)

- Skrócone wartościowanie
- Pętle
  - deklaracja zmiennej w pętli **`for`**
- Funkcje
  - przeciążanie funkcji

Na podstawie:

- A. Allain, *Przewodnik dla początkujących C++*
- S. Prata, *Szkoła programowania C++*

Politechnika Wrocławska

Pierwszy program w C++

Start here | helloworld.cpp

```
1  /*hellowowrld.cpp*/
2
3  #include<iostream>
4  /*Zwróć uwagę na inny plik nagłówkowy
5  niż używany w C stdio.h*/
6
7  using namespace std;
8  /*Instrukcja ułatwia posługiwanie się
9  krótszymi wersjami niektórych procedur.
10 W dalszej części kursu rozwiniemy
11 to wyjaśnienie.*/
12
13 int main(){
14     cout << "Hello world!\n";
15     /*W tym miejscu posługujemy się obiektem cout - strumieniem wyjściowym.
16     Wykorzystując operator wstawiania (<<), wstawiamy łańuch znakowy
17     do strumienia, czyli wypisujemy go na ekran.*/
18
19     /*jeżeli w funkcji main() nie ma instrukcji return,
20     wtedy domyślnie zwracana jest wartość 0.*/
21 }
22
```

Politechnika Wrocławska

Interakcja z użytkownikiem

Start here | read_number.cpp

```cpp
/*read_number.cpp*/

#include<iostream>

using namespace std;

int main(){
    int number;                    //deklaracja zmiennej typu całkowitego

    cout << "Podaj liczbe: ";

    cin >> number;
    //wczytanie danych ze strumienia wejściowego do zmiennej number

    cout << "Wprowadzona liczba to: " << number << endl;
    /*wyświetlenie wczytanych danych na ekranie
    Zwróć uwagę na wielokrotne powtórzenie operatora wstawiania,
    co powoduje wypisanie sekwencyjne danych.*/
}

```

Politechnika Wrocławska

Interakcja z użytkownikiem

- **<iostream>** - standard input/output streams library
- interakcja z użytkownikiem odbywa się z wykorzystaniem dwóch strumieni (obiektów) **cout** oraz **cin**
- **<ios>** - Input-Output base classes
- wybrane manipulatory zdefiniowane w <ios>
  - **left**
  - **right**
  - **scientific**

Politechnika Wrocławska

Interakcja z użytkownikiem

- **`<iomanip>`** - IO manipulators
- wybrane możliwości formatowania:
  - setbase
  - setfill
  - setpresicion
  - setw

Politechnika Wrocławska

Interakcja z użytkownikiem

Start here | iomanip_example.cpp

```cpp
/*iomanip_example.cpp
  na podstawie: http://www.cplusplus.com*/

#include<iostream>
#include<iomanip>
/*Biblioteka iomanip dostarcza następujące modifikatory:
  - setbase,
  - setfill,
  - setprecision,
  - setw.
*/

using namespace std;

int main(){
    cout << setbase(16);  //setbase zmienia podstawę systemu w jakim są wypisywane liczby
    cout << 110 << endl;  //liczba 110 wypisana jako szesnastkowe 6e
    cout << oct;          //podobny efekt można uzyskać wstawiając do strumienia: dec, hex, oct
    cout << 110 << endl;  //liczba 110 wypisana jako ósemkowe 156

```

Politechnika Wrocławska

Interakcja z użytkownikiem

Start here | iomanip_example.cpp

```
21      cout << setfill('x'); //setfill ustawia znak jakim są wypełniane pola
22      cout << setw (10);    //setw ustawia szerokość pola dla następnej danej
23      cout << 77 << endl;   //77 wypisane jako xxxxxxx115
24      // szerokość pola 10
25      // zapis w systemie ósemkowym to 115
26      // pole jest wypełnione znakami 'x'
27
28      cout.fill('_');       //podobny efekt jak setfill można uzyskać metodą fill() obiektu cout
29      cout << setw (10);
30      cout << 77 << endl;   //77 wypisane jako _______115
31
32      double f = 3.14159;
33      cout << setprecision(5) << f << endl; //setprecision ustawia liczbę cyfr znaczących
34      cout << setprecision(9) << f << endl;
35      cout.precision(3);                    //podobny efekt można uzyskać metodą precision() obiektu cout
36      cout << f << endl;
37      cout.precision(12);
38      cout << f << endl;
39  }
```

Politechnika Wrocławska

Typ `string`

- **`<string>`** definuje typ (klasę) **`string`**
- porównywanie stringów może być wykonane z użyciem operatorów logicznych
- operator '+' można wykorzystać do konkatenowania łańcuchów

Politechnika Wrocławska

Typ string

Start here | **string_name.cpp** | string_compare.cpp | string_concat.cpp

```cpp
/*string_name.cpp*/
/*Program pokazujący możliwość wykorzystania typu string*/

#include<iostream>
#include<string>
/*Aby korzystać z typu string, należy dołączyć bibliotekę string.
W odróżnieniu od typów wbudowanych typ string, nie jest dostępny.*/

using namespace std;

int main(){
    string name;          //deklaracja typu

    cout << "Podaj swoje imie: ";
    cin >> name;
    cout << "Czesc " << name << "!" << endl;
}

```

Politechnika Wrocławska

Typ string

Start here | string_name.cpp | **string_compare.cpp** | string_concat.cpp

```
1  /*string_compare.cpp*/
2  /*Program pokazujący możliwość porównywania łańcuchów.*/
3
4  #include<iostream>
5  #include<string>
6
7  using namespace std;
8
9  int main(){
10     string password;
11
12     cout << "Podaj haslo: ";
13     getline(cin, password, '\n');
14
15     if( password == "olsah" ){
16         cout << "Dostep przyznany" << endl;
17     }
18     else{
19         cout << "Nieprawidlowe haslo. Odmowa dostępu!" << endl;
20     }
21 }
22
```

Politechnika Wrocławska

Typ string

Start here | string_name.cpp | string_compare.cpp | **string_concat.cpp**

```cpp
/*string_concat.cpp*/
/*Program pokazujący możliwość łączenia łańcuchów.*/

#include<iostream>
#include<string>

using namespace std;

int main(){
    string name;
    string surname;

    cout << "Podaj swoje imie: ";
    cin >> name;
    cout << "Podaj swoje nazwisko: ";
    cin >> surname;

    string full_name = name + " " + surname;
    /*Operator + pozwala konkatenować (łączyć) łańcuchy
    trzy łańcuchy: name, " " (spacja) oraz surname,
    łączone są w jeden łańcuch.*/

    cout << "Nazywasz sie: " << full_name << endl;
}
```

Politechnika Wrocławska

Typ `bool`

- do przechowywania wartości logicznych można wykorzystać typ **`bool`**

Politechnika Wrocławska

Typ bool

Start here | bool_example.cpp

```
1  /*bool_example.cpp*/
2  /*Program pokazujący typ bool.*/
3
4  #include<iostream>
5
6  using namespace std;
7
8  int main(){
9      int x;
10
11     cout << "Podaj liczbe: ";
12     cin >> x;
13
14     bool x_equals_two = x == 2;
15     /*Wynik porównania x == 2 jest przypisywany
16     do zmiennej x_equals_two typu logicznego (bool).*/
17
18     /*Dalsze instrukcje wykonywane w zależności
19     od wartości zmiennej x_equals_two.*/
20     if( x_equals_two )
21         cout << "x = 2" << endl;
22 }
23
```

Politechnika Wrocławska

Skrócone wartościowanie

- Skrócone wartościowanie (short-circuit evaluation) jest to sposób wartościowania formuł logicznych, w którym nie obliczane są „zbędne” wyrażenia
- W przypadku koniunkcji **`a && b`** – jeżeli **`a`** jest fałszywe, to nie obliczaj **`b`**, bo wiadomo, że **`a && b`** jest fałszywe.
- W przypadku alternatywy **`a || b`** – jeżeli **`a`** jest prawdziwe, to nie obliczaj b, bo wiadomo, że **`a || b`** jest prawdziwe.

Politechnika Wrocławska

Skrócone wartościowanie

Start here | short_circuit_evaluation.cpp

```
1  /*short_circuit_evaluation.cpp*/
2  /*Program ilustruje wykorzystanie
3  skróconego wartościowania przy obliczaniu
4  wartości logicznej warunku*/
5
6  #include<iostream>
7
8  using namespace std;
9
10 int main(){
11     float x;
12
13     cout << "Podaj x: ";
14     cin >> x;
15
16     /*Prawdziwość drugiego zdania koniunkcji (5/x < 1)
17     jest sprawdzana tylko wtedy, gdy pierwsze zdanie (x!=0)
18     jest prawdziwe.
19     Jeżeli pierwsze zdanie jest fałszywe to całe wyrażenie
20     jest fałszywe niezależnie od oceny drugiego zdania.
21
22     W tym przypadku jeśli x == 0 nie jest wykonywane działanie 5/x.
23
24     Uruchom program podając różne wartości x, np.: 0, 2, -2, 8.*/
25     if( x!=0 && 5/x < 1 ){
26         cout << "5/x = " << 5/x;
27     }
28
29 }
30
```

Politechnika Wrocławska

Pętle

- `while`
- `for`
- `do-while`

Pętle

Deklaracja zmiennej w pętli `for`

Start here | for_example.cpp

```
1  /*for_example.cpp*/
2  /*Program pokazujący możliwość deklarowania zmiennej w pętli for.*/
3
4  #include<iostream>
5
6  using namespace std;
7
8  int main(){
9      for(int i=0; i<10; i++){
10         //Zwróć uwagę, że zmienna i jest deklarowana w instrukcji inicjalizacji.
11         cout << i << endl;
12     }
13     /*Odwołanie się do zmiennej i poza pętlą nie jest możliwe,
14     ze względu na ogranicznie zasięgu zmiennej.*/
15 }
16
```

Politechnika Wrocławska

Funkcje

Przeciążanie funkcji

- Mechanizm przeciążania funkcji umożliwia utworzenie więcej niż jednej definicji funkcji o tej samej nazwie, jeśli różnią się listą parametrów.
- Nie należy tej możliwości nadużywać.
- Przeciążanie (przeładowanie) ma sens, gdy dwie funkcji robią to samo, ale używając różnych argumentów.

Funkcje

Przeciążanie funkcji

Start here | function_overloading_triangle_area.cpp

```
1  /*function_overloading_triangle_area.cpp*/
2  /*Program pokazujący możliwość przeciążania funkcji
3  na przykładzie funkcji obliczających pole trójkąta.*/
4
5  #include<iostream>
6  #include<cmath> //sqrt()
7
8  using namespace std;
9
10 /*Deklaracja funkcji triangle_area().
11   Zwróć uwagę na "podwójną" deklarację funkcji.
12   Deklaracje różnią się różną listą argumentów:
13   - pierwsza oblicza pole wykorzystując podstawę a oraz wysokość h,
14   - druga oblicza pole wykorzystując długości boków: a, b, c.*/
15 double triangle_area(double a, double h);
16 double triangle_area(double a, double b, double c);
17
```

Funkcje

Przeciążanie funkcji

Start here | function_overloading_triangle_area.cpp

```
18  int main(){
19      //deklaracja wykorzystywanych zmiennych
20      double a, b, c, h;
21      a = 4;
22      h = 3;
23      b = 3;
24      c = 5;
25      //wyświetlenie wyników
26      cout << "Pole trojkata o podstawie " << a;
27      cout << " oraz wysokosci " << h << " = ";
28      cout << triangle_area(a,h) << "." << endl;
29
30      cout << "Pole trojkata o bokach " << a << ", ";
31      cout << b << ", " << c << " = ";
32      cout << triangle_area(a,b,c) << "." << endl;
33  }
```

Funkcje

Przeciążanie funkcji

Start here | function_overloading_triangle_area.cpp

```
35  //Funkcja oblicza pole wykorzystując podstawę a oraz wysokość h.
36  double triangle_area(double a, double h){
37      return a*h/2.;
38  }
39
40  /*Funkcja oblicza pole wykorzystując długości boków a, b, c.
41    Implementuje wzór Herona.*/
42  double triangle_area(double a, double b, double c){
43      double p = (a+b+c)/2.;
44      return sqrt(p*(p-a)*(p-b)*(p-c));
45  }
```

Politechnika Wrocławska

Podsumowanie

- Interakcja z użytkownikiem (`cout`, `cin`)
- Typ `string`
- Typ `bool`
- Skrócone wartościowanie
- Pętle
- Przeciążanie funkcji