# C++ (cz. 1)

CMI

Politechnika Warszawska
26.09.2020

Paweł Zawadzki

# O czym będzie?

- **O podstawach...**

# Kilka słów o języku

- **Bjarne Stroustrup (1979… 1985 r.);**
- **„C with Classes”;**
- **wieloparadygmatowy;**
- **ogólnego przeznaczenia;**
- **zachowuje zgodność z C;**
- **bezpośrednie zarządzanie pamięcią i dostęp do zasobów;**
- **standardy: C++98, C++03, C++11, C++14, C++17, C++20 (?), C++23 (?).**

# Środowisko pracy

- **Można korzystać z IDE (Code::Blocks, Dev-C++, CLion itd.);**
- **Można korzystać z konsoli i „ręcznie” robić to, co robi za nas IDE.**

# Hello, World!

```cpp
#include <iostream>

using namespace std;

int main(int argc, char *argv[]) {

    cout << "Hello, World!" << endl;

    return 0;
}
```

```c
#include <stdio.h>

int main(int argc, char *argv[]) {

    printf("Hello, World!\n");

    return 0;
}
```

# Komentarze

```
#include <iostream>

using namespace std;

/*
 * Czasami chcemy napisać komentarz nad funkcją,
 * aby wyjaśnić, do czego ona służy.
 * Ale nie powinniśmy próbować opisywać w komentarzu,
 * jak ten kod działa.
 */
int main(int argc, char *argv[]) {

    string challenge_text = "Pokaż, że potrafisz inaczej.";
    cout << challenge_text << "\n";
//    printf("%s\n", challenge_text);

    return 0;
}
```

# Typy danych (liczba całkowita)

```
/* Liczba całkowita: short, int, long, long long */

// (...)
int daysInYear = 365;
int daysInWeek = 7;

int result_1 = daysInYear / daysInWeek;

cout << "Result 1: " << result_1 << "\n";
// (...)
```

# Typy danych (liczba rzeczywista)

```
/* Liczba rzeczywista: float, double, long double  */

// (...)
float nearTwo = 1.999;

int floor = nearTwo;

cout << "floor: " << floor << "\n";
// (...)
```

# Typy danych (wartość logiczna)

```
/* Wartość logiczna: true (1), false (0) */

// (...)
bool raining = 1;
bool sunny = true;
bool warm = false;

cout << "Is raining? " << raining << "\n";
cout << "Is sunny? " << sunny << "\n";
cout << "Is warm outside? " << warm << "\n";
// (...)
```

# Typy danych (typ znakowy)

```
/* Typ znakowy: char */

// (...)
char letter_a = 'a';
char letter_B = 'B';

cout << "Letter a: " << letter_a << "\n";
cout << "Letter B: " << letter_B << "\n";
// (...)
```

**Sprawdź:** tablica kodów ASCII

(https://pl.wikipedia.org/wiki/ASCII)

# Typy danych (łańcuch znaków)

```
/* Łańcuch znaków: string, char* */

// (...)
string declaration = "Kocham CMI (szczególnie w sobotę).";

cout << "Sentece \"" << declaration;

cout << "\" with length: " << declaration.length() << "." << endl;


/*char* second_declaration = "Kocham CMI (najbardziej w
niedzielę).";
printf("%s\n", second_declaration);*/

// (...)
```

# Typy danych (wartość stała)

```
/* Wartość stała: const */

// (...)
const int myAge = 29;

myAge = 30; // Niedozwolone nadpisywanie wartości stałej.

cout << "My age: " << myAge << endl;
// (...)
```

# Typy danych (tablice)

```
/* Tablice jednowymiarowe */

// (...)
int points[] = {20, 99, 30, 60, 66, 55, 72, 19, 79, 70, 89, 11};
int n = sizeof(points) / sizeof(points[0]);

cout << "N of points: " << n << "\n";
cout << "First point: " << points[0] << "\n";
cout << "Last point: " << points[n - 1] << "\n";

points[1] = 98;
cout << "Points[1] = " << points[1] << "\n";
// (...)
```

# Typy danych (tablice 2D)

```
/* Tablice dwuwymiarowe
 * jeśli nie są dynamicznie alokowane,
 * wymagają podania drugiego wymiaru.
 */
// (...)
int points2d[][3] = {{1, 2, 3},
                     {4, 5, 6},
                     {7, 8, 9}};

Points2d[0][1] = 10;
// (...)
```

# Typy danych (struktury)

```
/* Struktura: struct */

struct Grantobiorca_CMI {
    string name;
    string surname;
    int age;
    bool sex;
    double points[4];
};

int main(int argc, char *argv[]) {
    // (...)
    Grantobiorca_CMI cmi_1 = {"Jan", "Kowalski", 18, 1, {3.2, 2, 1}};
    Grantobiorca_CMI cmi_2 = {"Jarosław", "Budka", 36, 1, {15, 12, 15, 14}};

    cmi_1.surname = "Kowalewski";

    cout << "Imię: " << cmi_1.name << "\n";
    cout << "Nazwisko: " << cmi_1.surname << "\n";
```

# Zmienne i wskaźniki (na zmienne)

```
/* Zmienne (x, y) bez inicjalizacji podczas deklaracji.
 * Pamiętamy o deklarowaniu zmiennej (jej typu) przed jej użyciem. */
// (...)
int x, y, area;
int* xPtr;

x = 10;
y = 6;
xPtr = &x;

area = x * y;
cout << "*xPtr = " << *xPtr << "\n";
*xPtr = 20;
cout << "x = " << x << "\n";

cout << area << "\n";
cout << &area << "\n";
```

# Rzutowanie i promocja typów

```
/* Rzutowanie i promocja typów */
// (...)
float squareArea = 1.44;
int triangleArea = 9;

float sum = squareArea + triangleArea;
cout << squareArea << " + " << triangleArea << " = " << sum << "\n";

int distance = 121;
int step = 3;
//double numOfSteps = (double) distance / step;
double numOfSteps = distance / step;

cout << distance << " / " << step << " = " << numOfSteps << "\n";
// (...)
```

# Operatory i funkcje matematyczne

```
/* Aby skorzystać z funkcji matematycznych (min, max, pow,
sqrt, sin, cos itp.), należy załączyć bibliotekę cmath. */

#include <cmath>
int main(int argc, char* argv[]){
    // (...)
    sum = x + y;
    difference = x - y;
    product = x * y;
    quotient = x / y;
    power = pow(x, 2);
    squareRoot = sqrt(x);
    modulo = p % q;

    x++;
    y--;
    ++x;
    --y;
    // (...)
```

# Operatory porównania (==, >, <, !=, >=, <=)

```
// (...)
int x = 5, y = 7;
double p = 5, q = 7;
string a = "Ala", b = "Kot", ola = "Ola";

cout << "x == y (?): " << (x == y) << "\n";
cout << "p == q (?): " << (p == q) << "\n";
cout << "a == b (?): " << (a == b) << "\n";

cout << "x == p (?): " << (x == p) << "\n";

cout << "x > y (?): " << (x>y) << "\n";
cout << "q > p (?): " << (q>p) << "\n";
cout << "a <= b (?): " << (a<=b) << "\n";

cout << "x != y (?): " << (x!=y) << "\n";
cout << "ola != ola (?): " << (ola!=ola) << "\n";
// (...)
```

# Operatory logiczne (&&, ||, !)

```
/* Operatory logiczne: and, or, not */

// (...)
int x = 5, y = 7;
string ola = "Ola", ala = "Ala";

cout << "(x < y) && (x+y > 2*x) (?) ";
cout << ((x < y) && (x + y > 2 * x)) << "\n";

cout << "(x < y) || !(ola == ala) (?) ";
cout << ((x < y) || !(ola == ala)) << "\n";
// (...)
```

# Komendy sterujące

- **Instrukcja warunkowa: if, if else;**
- **pętla for;**
- **pętla while (ewentualnie: do{}while()).**

```
int n = 5;
string names[n] = {"Zuza", "Radek", "Marek", "Michał", "Paweł"};

for (int i = 0; i < n; i++) {
    if (i < n - 1 && names[i] < names[i + 1]) {
        cout << "Name: " << names[i] << "\n";
        cout << "Name: " << names[i] + names[i + 1] << "\n";
    }
}
```

# Alokowanie (i zwalnianie) pamięci

- **Kiedy alokujemy pamięć?**
  - Gdy nie znamy rozmiaru tablicy w trakcie pisania kodu;
  - gdy rozmiar tablicy zmienia się w trakcie działania.
- **Kiedy zwalniamy pamięć?**
  - Gdy ją wcześniej „ręcznie” alokowaliśmy i nie jest nam już dłużej potrzebna.

# Alokowanie (i zwalnianie) pamięci

```
// (...)
double *points;
int nElems = 5;
points = new(nothrow) double[nElems];

if (points == NULL) {
    cout << "Null pointer has occured\n" << "\n";
    return -1;
}

points[0] = 1;
points[1] = 5;
points[2] = 3;
points[3] = 4;
points[4] = 5;

delete[] points;
// (...)
```