Podstawy Programowania
Semestr letni 2022/23
Materiały z laboratorium i zadania domowe

Przemysław Olbratowski

21 marca 2023

Slajdy z wykładu są dostępne w serwisie UBI. Informacje organizacyjne oraz formularz do uploadu prac domowych znajdują się na stronie `info.wsisiz.edu.pl/~olbratow`. Przy zadaniach domowych w nawiasach są podane terminy sprawdzeń.

4.2 Zadania domowe z działu Funkcje (29 marca, 5, 26 kwietnia)

4.2.1 Bernoulli: Doświadczenie Bernoulliego

Napisz funkcję `bernoulli`, która przyjmuje liczbę rzeczywistą p z przedziału od 0 do 1 włącznie i zwraca prawdę z prawdopodobieństwem p albo fałsz z prawdopodobieństwem $1 - p$. Funkcja powinna być przystosowana do użycia w przykładowym programie poniżej. Funkcja korzysta tylko z pliku nagłówkowego `cstdlib`.

Przykładowy program

```
int main() {
    std::srand(std::time(nullptr));
    std::cout << std::boolalpha;
    for (int counter = 0; counter < 10; ++counter) {
        std::cout << bernoulli(0.2) << " "; }
    std::cout << std::endl; }
```

Przykładowe wykonanie

```
Out: false false false true false false true false true false
```

4.2.2 Choose: Wybór zmiennej

Napisz funkcję `choose`, która przyjmuje wartość logiczną oraz modyfikujące referencje dwóch zmiennych rzeczywistych i zwraca modyfikującą referencję pierwszej albo drugiej z nich jeżeli przekazana wartość logiczna jest odpowiednio prawdziwa albo fałszywa. Funkcja powinna być przystosowana do użycia w przykładowym programie poniżej. Funkcja nie korzysta z żadnych plików nagłówkowych.

Przykładowy program

```
int main() {
    double a = 2.3, b = 3.2;
    choose(a, b, false) = 10.9;
    std::cout << a << " " << b << std::endl; }
```

Wykonanie

```
Out: 2.3 10.9
```

4.2.3 Coordinates: Współrzędne biegunowe i kartezjańskie

Rozważmy punkt na płaszczyznie. Napisz funkcję `coordinates`, która przyjmuje jego współrzędne biegunowe z kątem w radianach oraz modyfikujące referencje dwóch zmiennych rzeczywistych i wpisuje do tych zmiennych współrzędne kartezjańskie tego punktu. Funkcja powinna być przystosowana do użycia w przykładowym programie poniżej. Funkcja korzysta tylko z pliku nagłówkowego `cmath`.

Przykładowy program

```
constexpr double pi = 4 * std::atan(1.);

int main() {
    double x, y;
    coordinates(x, y, 1., 60. * pi / 180.);
    std::cout << x << " " << y << std::endl; }
```

Wykonanie

```
Out: 0.5 0.866025
```

4.2.4 Digits: Cyfry dziesiętne

Napisz funkcję `digits`, która przyjmuje nieujemną liczbę całkowitą i zwraca liczbę cyfr w jej zapisie dziesiętnym. Napisz funkcję `digit`, która przyjmuje nieujemną liczbę całkowitą oraz numer cyfry w jej zapsie dziesiętnym i zwraca tę cyfrę. Cyfry numerujemy od zera dla cyfry jedności. Funkcje powinny być przystosowane do użycia w przykładowym programie poniżej. Funkcje nie korzystają z żadnych plików nagłówkowych.

Przykładowy program

```
int main() {
    std::cout << digits(2018) << " " << digit(2018, 2) << std::endl; }
```

Wykonanie

```
Out: 4 0
```

4.2.5 DMS: Stopnie, minuty, sekundy

W geografii kąty często wyraża się podając całkowite liczby stopni, minut i sekund. Napisz funkcję `dms`, która przyjmuje rzeczywistą liczbę stopni oraz modyfikujące referencje trzech zmiennych całkowitych i wpisuje do tych zmiennych liczby stopni, minut oraz sekund. Funkcja powinna być przystosowana do użycia w przykładowym programie poniżej. Funkcja nie korzysta z żadnych plików nagłówkowych.

Przykładowy program

```
int main() {
    int degrees, minutes, seconds;
    dms(degrees, minutes, seconds, 123.37);
    std::cout << degrees << " " << minutes << " " << seconds << std::endl; }
```

Wykonanie

```
Out: 123 22 12
```

4.2.6 Euclid: Algorytm Euklidesa

Najmniejszą wspólną wielokrotność dwóch dodatnich liczb całkowitych można obliczyć dzieląc ich iloczyn przez największy wspólny dzielnik. Napisz funkcję `euclid`, która przyjmuje modyfikujące referencje dwóch zmiennych całkowitych zawierających wyjściowe liczby i wpisuje do tych zmiennych największy wspólny dzielnik i najmniejszą wspólną wielokrotność tych liczb. Funkcja powinna być przystosowana do użycia w przykładowym programie poniżej. Funkcja nie korzysta z żadnych plików nagłówkowych.

Przykładowy program

```
int main() {
    int a = 60, b = 1240;
    euclid(a, b);
    std::cout << a << " " << b << std::endl; }
```

Wykonanie

```
Out: 20 3720
```

4.2.7 Exchange: Zamiana wartości zmiennej

Napisz funkcję `exchange`, która przyjmuje liczbę rzeczywistą oraz modyfikującą referencję zmiennej rzeczywistej, wpisuje liczbę do tej zmiennej i zwraca poprzednią wartość zmiennej. Funkcja powinna być przystosowana do użycia w przykładowym programie poniżej. Funkcja nie korzysta z żadnych plików nagłówkowych.

Przykładowy program

```
int main() {
    double a = 2.71828, b = exchange(a, 3.14159);
    std::cout << a << " " << b << std::endl; }
```

Wykonanie

```
Out: 3.14159 2.71828
```

4.2.8 Factorial: Silnia

Napisz funkcję `factorial`, która przyjmuje nieujemną liczbę całkowitą i zwraca jej silnię. Funkcja powinna być przystosowana do użycia w przykładowym programie poniżej. Funkcja nie korzysta z żadnych plików nagłówkowych.

Przykładowy program

```
int main() {
    std::cout << factorial(6) << std::endl; }
```

Wykonanie

```
Out: 720
```

4.2.9 Fraction: Część całkowita i ułamkowa - grupowo

Napisz funkcję `fraction`, która przyjmuje nieujemną liczbę rzeczywistą oraz modyfikujące referencje dwóch zmiennych rzeczywistych i wpisuje do tych zmiennych całkowitą oraz ułamkową część podanej liczby. Funkcja powinna być przystosowana do użycia w przykładowym programie poniżej. Funkcja korzysta tylko z pliku nagłówkowego `cmath`.

Przykładowy program

```
int main() {
    double integral, fractional;
    fraction(integral, fractional, 3.14159);
    std::cout << integral << " " << fractional << std::endl; }
```

Wykonanie

```
Out: 3 0.14159
```

4.2.10 Geometric: Ciąg geometryczny

n-ty wyraz ciągu geometrycznego o wyrazie początkowym a_0 i ilorazie q jest równy $a_n = a_0 q^n$. Napisz funkcję `geometric`, która przyjmuje a_0, q oraz n i zwraca a_n. Funkcja powinna być przystosowana do użycia w przykładowym programie poniżej. Funkcja korzysta tylko z pliku nagłówkowego `cmath`.

Przykładowy program

```
int main() {
    std::cout << geometric(8., -0.5, 3) << std::endl; }
```

Wykonanie

```
Out: -1
```

4.2.11 Implication: Implikacja

Napisz funkcję `implication`, która przyjmuje wartości logiczne p oraz q i zwraca $p \Rightarrow q$. Funkcja powinna być przystosowana do użycia w przykładowym programie poniżej. Funkcja nie używa instrukcji wyboru, instrukcji warunkowej ani operatora warunkowego oraz nie korzysta z żadnych plików nagłówkowych.

Przykładowy program

```
int main() {
    std::cout << std::boolalpha << implication(true, false) << std::endl; }
```

Wykonanie

```
Out: false
```

Wskazówka $(p \Rightarrow q) \Leftrightarrow (\neg p \vee q)$

4.2.12 Less: Mniejsza z dwóch liczb

Napisz funkcję `less`, która przyjmuje dwie liczby rzeczywiste i zwraca mniejszą z nich. Funkcja powinna być przystosowana do użycia w przykładowym programie poniżej. Funkcja nie używa instrukcji warunkowej i nie korzysta z żadnych plików nagłówkowych.

Przykładowy program

```
int main() {
    std::cout << less(3.12, 2.13) << std::endl; }
```

Wykonanie

```
Out: 2.13
```

4.2.13 Pitagoras: Trójki pitagorejskie

Trójką pitagorejską nazywamy każde trzy dodatnie liczby całkowite a, b, c, takie że $a^2 + b^2 = c^2$. Trójkę nazywamy pierwotną, jeżeli liczby a, b, c są względnie pierwsze. Napisz program `pitagoras`, który wczytuje ze standardowego wejścia dodatnią liczbę całkowitą d i wypisuje na standardowe wyjście wszystkie pierwotne trójki pitagorejskie, dla których $c < d$. Program załącza tylko plik nagłówkowy `iostream`.

Przykładowe wykonanie

```
 In: 30
Out: 3 4 5
Out: 5 12 13
Out: 8 15 17
Out: 7 24 25
Out: 20 21 29
```

4.2.14 Power: Potęga całkowita

Napisz funkcję `power`, która przyjmuje niezerową liczbę rzeczywistą x oraz dowolną liczbę całkowitą n i zwraca n-tą potęgę liczby x. Funkcja powinna być przystosowana do użycia w przykładowym programie poniżej. Funkcja nie używa instrukcji warunkowej i nie korzysta z żadnych plików nagłówkowych.

Przykładowy program

```
int main() {
    std::cout << power(-0.5, -3) << std::endl; }
```

Wykonanie

```
Out: -8
```

4.2.15 Prime: Czy liczba jest pierwsza - indywidualnie

Pierwszość liczby najprościej sprawdzić badając jej podzielność przez wszystkie liczby większe od jednego i mniejsze od niej samej. Napisz funkcję `prime`, która przyjmuje nieujemną liczbę całkowitą i zwraca prawdę jeśli jest ona pierwsza oraz fałsz w przeciwnym razie. Funkcja powinna być przystosowana do użycia w przykładowym programie poniżej. Funkcja nie używa instrukcji warunkowej i nie korzysta z żadnych plików nagłówkowych.

Przykładowy program

```
int main() {
    std::cout << std::boolalpha << prime(97) << std::endl; }
```

Wykonanie

```
Out: true
```

4.2.16 RNG: Generator liczb pseudolosowych

Rozważmy prosty generator liczb pseudolosowych, który wyznacza kolejną liczbę x_{next} na podstawie poprzedniej x_{previous} według wzoru

$$x_{\text{next}} = (33 * x_{\text{previous}} + 1) \bmod 1024$$

Napisz bezargumentową funkcję `rng`, która zwraca modyfikującą referencję statycznej zmiennej zadeklarowanej wewnątrz funkcji. Każde wywołanie funkcji traktuje wartość tej zmiennej jako x_{previous} i nadaje jej wartość x_{next}. Funkcja powinna być przystosowana do użycia w przykładowym programie poniżej. Funkcja nie korzysta z żadnych plików nagłówkowych.

Przykładowy program

```
int main() {
    rng() = 7;
    for (int counter = 0; counter < 10; ++counter) {
        std::cout << rng() << " "; }
    std::cout << std::endl; }
```

Wykonanie

```
Out: 232 489 778 75 428 813 206 655 112 625
```

4.2.17 Root: Pierwiastek kwadratowy

Pierwiastek kwadratowy z liczby rzeczywistej x można obliczyć następująco. Jeżeli $x < 1$ to pierwiastek leży między 0 a 1, zaś w przeciwnym razie między 1 a x. Bierzemy środek r odpowiedniego z tych przedziałów. Jeżeli $x < r^2$, to poszukiwany pierwiastek leży w lewej połowie przedziału, zaś w przeciwnym razie leży w prawej połowie. Do dalszych rozważań bierzemy więc odpowiednią połowę, dzielimy ją na pół i tak dalej. Dzięki temu w każdym kroku dwukrotnie zawężamy przedział, w którym leży pierwiastek. Ze względu na skończoną dokładność obliczeń środek któregoś kolejnego przedziału okaże się numerycznie równy jednemu z jego krańców. Oznacza to, że znaleźliśmy wynik z dokładnością maszynową. Napisz funkcję `root`, która przyjmuje nieujemną liczbę rzeczywistą i zwraca jej pierwiastek obliczony opisaną metodą. Funkcja powinna być przystosowana do użycia w przykładowym programie poniżej. Funkcja nie korzysta z żadnych plików nagłówkowych.

Przykładowy program

```
int main() {
    std:: cout << root(7) << " " << std::sqrt(7) << std::endl; }
```

Wykonanie

```
Out: 2.64575 2.64575
```

4.2.18 Round: Zaokrąglanie z zadaną dokładnością

Zdefiniowana w nagłówku `cmath` funkcja `std::round` zaokrągla podaną liczbę rzeczywistą do najbliższej liczby całkowitej, na przykład `std::round(3.14159)` daje w wyniku 3. Napisz ulepszoną funkcję `round`, która przyjmuje liczbę rzeczywistą oraz liczbę cyfr po przecinku i zwraca podaną liczbę rzeczywistą zaokrągloną do podanej liczby cyfr po przecinku. Funkcja powinna być przystosowana do użycia w przykładowym programie poniżej. Funkcja korzysta tylko z pliku nagłówkowego `cmath`.

Przykładowy program

```
int main() {
    std::cout << round(3.14159, 3) << std::endl; }
```

Wykonanie

```
Out: 3.142
```

4.2.19 Sign: Znak liczby całkowitej

Napisz funkcję `sign`, która przyjmuje liczbę całkowitą i zwraca -1, 0 lub 1 jeżeli liczba ta jest odpowiednio ujemna, równa zeru lub dodatnia. Funkcja powinna być przystosowana do użycia w przykładowym programie poniżej. Funkcja nie używa instrukcji warunkowej ani operatora warunkowego i nie korzysta z żadnych plików nagłówkowych.

Przykładowy program

```
int main() {
    std::cout << sign(-15) << std::endl; }
```

Wykonanie

```
Out: -1
```

4.2.20 Swap: Zamiana wartości dwóch zmiennych

Napisz funkcję `swap`, która przyjmuje modyfikujące referencje dwóch zmiennych rzeczywistych i zamienia wartości tych zmiennych. Funkcja powinna być przystosowana do użycia w przykładowym programie poniżej. Funkcja nie korzysta z żadnych plików nagłówkowych.

Przykładowy program

```
int main() {
    double a = 2.71828, b = 3.14159;
    swap(a, b);
    std::cout << a << " " << b << std::endl; }
```

Wykonanie

```
Out: 3.14159 2.71828
```

4.2.21 Uniform: Płaski rozkład prawdopodobieństwa

Napisz funkcję `uniform`, która przyjmuje liczby rzeczywiste a oraz b i zwraca losową liczbę rzeczywistą z przedziału domkniętego od a do b. Funkcja powinna być przystosowana do użycia w przykładowym programie poniżej. Funkcja korzysta tylko z pliku nagłówkowego `cstdlib`.

Przykładowy program

```
int main() {
    std::srand(std::time(nullptr));
    for (int counter = 0; counter < 10; ++counter) {
        std::cout << uniform(-5., 10.) << " "; }
    std::cout << std::endl; }
```

Przykładowe wykonanie

```
Out: -3.39 9.39528 -4.18653 -4.95605 0.977203 -2.45064 1.47938 -4.03088 9.50011 -2.12058
```