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

Przemysław Olbratowski

8 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.

2.2 Zadania domowe z działu Pętle (15, 22, 29 marca)

2.2.1 Bifactorial: Podwójna silnia

Podwójna silnia nieujemnej liczby całkowitej n, oznaczana jako $n!!$, to iloczyn wszystkich dodatnich liczb całkowitych mniejszych lub równych n, o takiej samej parzystości jak n, przy czym $0!! = 1$. Napisz program `bifactorial`, który wczytuje ze standardowego wejścia nieujemną liczbę całkowitą i wypisuje na standardowe wyjście jej podwójną silnię. Program załącza tylko plik nagłówkowy `iostream`.

Przykładowe wykonanie

```
 In: 6
Out: 48
```

2.2.2 Binary: Wyszukiwanie binarne

Napisz program `guess` odgadujący pomyślaną przez użytkownika liczbę. Przed uruchomieniem użytkownik wybiera losowo liczbę całkowitą z przedziału od zera włącznie do stu wyłącznie. Po uruchomieniu program wypisuje na standardowe wyjście pewną liczbę z tego przedziału i wczytuje ze standardowego wejścia odpowiedź użytkownika równą `-1`, `0`, lub `+1`. Odpowiedzi te oznaczają odpowiednio, że pomyślana liczba jest mniejsza, równa, lub większa od liczby wyświetlonej przez program. Program kontynuuje zgadywanie aż do odgadnięcia właściwej liczby. Program powinien odgadnąć tę liczbę w nie więcej niż siedmiu próbach. Program załącza tylko plik nagłówkowy `iostream`.

Przykładowe wykonanie

```
Out: 50
 In: -1
Out: 25
 In: -1
Out: 12
 In: 1
Out: 18
 In: -1
Out: 15
 In: 1
Out: 16
 In: 1
Out: 17
 In: 0
```

Wskazówka Zastosuj wyszukiwanie binarne. Pomyślana przez użytkownika liczba leży w przedziale od 0 do 100. Na początku program wyświetla liczbę leżącą w połowie tego przedziału, czyli 50. Jeżeli użytkownik odpowie -1, to jego liczba leży w przedziale od 0 do 50, i w następnej próbie program wyświetla liczbę w połowie tego przedziału, czyli 25. Jeżeli natomiast użytkownik odpowie +1, to jego liczba leży w przedziale od 50 do 100, i w następnej próbie program wyświetla liczbę w połowie tego przedziału, czyli 75. W ten sposób w każdej próbie program dwukrotnie zawęża przedział, w którym może leżeć pomyślana przez użytkownika liczba. Zapewnia to zgadnięcie liczby w nie więcej niż siedmiu próbach.

2.2.3 Clock: Zegar cyfrowy

Napisz program `clock`, który w przybliżeniu co sekundę wypisuje na standardowe wyjście godzinę sformatowaną jak poniżej, poczynając od północy. Program załącza tylko pliki nagłówkowe `iomanip` i `iostream`.

Początek przykładowego wykonania

```
Out: 00:00:00
Out: 00:00:01
Out: 00:00:02
```

Wskazówka Do odczekania sekundy użyj pustej pętli powtarzającej się dostatecznie dużo razy.

2.2.4 Collatz: Hipoteza Collatza

Jeżeli liczba jest parzysta, to dzielimy ją przez dwa, a jeśli jest nieparzysta, to mnożymy przez trzy i dodajemy jeden. Z otrzymanym wynikiem postępujemy tak samo. Hipoteza Collatza mówi, że w końcu zawsze otrzymamy jeden. Do tej pory nie wiadomo, czy to prawda. Napisz program `collatz`, który wczytuje ze standardowego wejścia dodatnią liczbę całkowitą i wypisuje na standardowe wyjście kolejne liczby otrzymane w opisany sposób, aż do jedynki. Program załącza tylko plik nagłówkowy `iostream`.

Przykładowe wykonanie

```
 In: 9
Out: 28 14 7 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
```

2.2.5 Fibonacci: Ciąg Fibonacciego - indywidualnie

Ciąg Fibonacciego zaczyna się od wyrazów 1 i 2, a każdy następny jest sumą dwóch poprzednich. Napisz program `fibonacci`, który wczytuje ze standardowego wejścia nieujemną liczbę całkowitą n i drukuje na standardowe wyjście n pierwszych wyrazów ciągu Fibonacciego. Program załącza tylko plik nagłówkowy `iostream`.

Przykładowe wykonanie

```
 In: 9
Out: 1 2 3 5 8 13 21 34 55
```

2.2.6 Loops: Pętle

Napisz program `loops`, który wczytuje ze standardowego wejścia nieujemną liczbę całkowitą n i wypisuje na standardowe wyjście w trzech kolejnych liniach wszystkie liczby niepodzielne przez 3 od 0 włącznie do n wyłącznie, w kolejności rosnącej; wszystkie liczby podzielne przez 3 od 0 włącznie do n wyłącznie, w kolejności rosnącej; oraz wszystkie liczby parzyste większe lub równe $-n$ i mniejsze lub równe n, w kolejności malejącej. Program zawiera tylko jedną instrukcję warunkową i załącza tylko plik nagłówkowy `iostream`.

Przykładowe wykonanie

```
 In: 10
Out: 1 2 4 5 7 8
Out: 0 3 6 9
Out: 10 8 6 4 2 0 -2 -4 -6 -8 -10
```

2.2.7 Nominals: Nominały

Napisz program `nominals`, który wczytuje ze standardowego wejścia nieujemną liczbę całkowitą i wypisuje na standardowe wyjście wszystkie mniejsze od niej liczby postaci 1, 2, 5, 10, 20, 50,... Program załącza tylko plik nagłówkowy `iostream`.

Przykładowe wykonanie

```
 In: 1000
Out: 1 2 5 10 20 50 100 200 500
```

2.2.8 Pi: Oszacowanie liczby pi metodą Monte-Carlo

Przybliżoną wartość liczby π można wyznaczyć następująco. Rozważmy kwadrat o boku 1 oraz zawartą w nim ćwiartkę koła o promieniu 1. Pola kwadratu i ćwiartki koła wynoszą odpowiednio $A_s = 1$ i $A_c = \pi/4$. Wybierzmy losowo dużo punktów tak, aby równomiernie wypełniały cały kwadrat. Stosunek liczby punktów wewnątrz ćwiartki koła do liczby wszystkich punktów w kwadracie jest w przybliżeniu równy stosunkowi pól tych figur, $N_c/N_s \approx A_c/A_s$. Dostajemy stąd

$$\pi \approx 4N_c/N_s$$

Napisz program `pi`, który wczytuje ze standardowego wejścia liczbę punktów do wylosowania i wypisuje na standardowe wyjście otrzymane przybliżenie liczby π. Program załącza tylko pliki nagłówkowe `cstdlib`, `ctime` i `iostream`.

Przykładowe wykonanie

```
 In: 10000
Out: 3.144
```

2.2.9 Polyhedron: Wielościany foremne

W każdym wielościanie foremnym całkowita liczba krawędzi, e_t, liczba krawędzi jednej ściany, e_f, oraz liczba krawędzi wychodzących z jednego wierzchołka, e_v, spełniają zależność

$$2e_t(e_f + e_v) = (2 + e_t)e_f e_v$$

Napisz program `polyhedron`, który wypisuje na standardowe wyjście wszystkie możliwe kombinacje całkowitej liczby ścian, $f_t = 2e_t/e_f$, i liczby krawędzi jednej ściany, e_f. Program załącza tylko plik nagłówkowy `iostream`.

Przykładowe wykonanie

```
Out: 4 3
Out: 6 4
Out: 8 3
Out: 12 5
Out: 20 3
```

2.2.10 Precision: Dokładność maszynowa

Dokładnością maszynową nazywamy najmniejszą potęgę dwójki, która dodana do jedynki daje wynik numerycznie różny od jednego. Pojęcie to odnosi się tylko do liczb zmiennoprzecinkowych i chodzi tu o potęgę z wykładnikiem ujemnym. Dokładność maszynowa jest różna dla zmiennych różnego typu. Napisz program `precision`, który doświadczalnie wyznacza i wypisuje na standardowe wyjście dokładności maszynowe typów `float`, `double`, oraz `long double`. Program załącza tylko plik nagłówkowy `iostream`.

Przykładowe wykonanie

```
Out: 5.960464e-08 1.110223e-16 5.421011e-20
```

Wskazówka Dokładność maszynową można wyznaczyć doświadczalnie w następujący sposób. Zaczynamy od zerowej potęgi dwójki równej jeden, a następnie rozpatrujemy coraz mniejsze potęgi. Każdą kolejną potęgę dodajemy do jedynki i sprawdzamy, czy wynik jest numerycznie równy jedności. Trzeba jednak pamiętać, że procesor nie wykonuje operacji zmiennoprzecinkowych bezpośrednio na zmiennych, lecz w swoich wewnętrznych rejestrach. Może się więc zdarzyć, że zamiast wyznaczyć dokładność żądanego typu znajdziemy dokładność rejestrów procesora, która jest na ogół większa. Aby temu zapobiec, wynik każdego dodawania należy najpierw wpisać do zmiennej żądanego typu i dopiero tę zmienną porównać z jedynką. Kolejną potęgę dwójki najprościej wyznaczyć dzieląc poprzednią potęgę przez dwa.

2.2.11 Race: Gra planszowa Chińczyk

Uproszczona gra Chińczyk przebiega następująco. Tor składa się z 41 pól numerowanych od 0 do 40. Gracze ustawiają się na polu 0. Pierwszy gracz rzuca kostką i przemieszcza się o tyle pól do przodu, ile oczek wyrzucił. Jeżeli staje na polu, na którym stoją już inni gracze, to gracze ci wracają na pole 0. Jeżeli gracz wyrzucił 6, to rzuca jeszcze raz. Jeżeli po wykonaniu ruchu gracz wyszedłby za pole 40, to stoi w miejscu. Wygrywa ten, kto jako pierwszy znajdzie się na polu 40. Napisz program `race` grający z użytkownikiem w taką grę. Po uruchomieniu program wypisuje na standardowe wyjście pozycje swoją i użytkownika. Następnie symuluje rzut kostką, wypisuje liczbę oczek i przemieszcza się. Potem wczytuje ze standardowego wejścia liczbę naturalną określającą, z jaką siłą użytkownik chce rzucić kostką. Tyle razy symuluje rzut kostką, jako liczbę oczek bierze ostatni wynik i wypisuje go. Jednocześnie odpowiednio przesuwa użytkownika. Po każdym ruchu ponownie wypisuje pozycję swoją i użytkownika. Program załącza tylko pliki nagłówkowe `cstdlib`, `ctime` i `iostream`.

Początek i koniec przykładowego wykonania

```
Out: 0 0
Out: 5
Out: 5 0
 In: 17
Out: 5
Out: 0 5
Out: 6
Out: 6 5
Out: 1
Out: 7 5
...
Out: 38 35
Out: 3
Out: 38 35
 In: 12
Out: 5
Out: 38 40
```

2.2.12 Section: Losowy podział liczby

Napisz program `section`, który wczytuje ze standardowego wejścia dodatnią liczbę całkowitą n, po czym wypisuje na standardowe wyjście n losowych liczb nieujemnych, które dają w sumie jeden. Program załącza tylko pliki nagłówkowe `cstdlib`, `ctime` oraz `iostream`.

Przykładowe wykonanie

```
 In: 3
Out: 0.54095 0.345354 0.113696
```

2.2.13 Stars: Wzorki z gwiazdek

Napisz program `stars`, który wczytuje ze standardowego wejścia nieujemną liczbę całkowitą n i wypisuje na standardowe wyjście wybrany wzorek z gwiazdek złożony z $2n+1$ wierszy i kolumn. Program załącza tylko pliki nagłówkowe `cstdlib` i `iostream`.

Przykładowe wzorki

```
   *     *******  *     *     *     *******     *
   *     *     *   *   *     * *     *****     ***
   *     *     *    * *     *   *     ***     *****
*******  *     *     *     *     *     *     *******
   *     *     *    * *     *   *     ***     *****
```

```
*     *     *   *   *     * *     *****     ***
*     *******  *     *     *     *******     *
```

2.2.14 Sum: Suma skończona

Napisz program `sum`, który wczytuje ze standardowego wejścia dodatnią liczbę całkowitą n i wypisuje na standardowe wyjście sumę

$$4\sum_{k=1}^{n}\frac{(-1)^{k+1}}{2k-1}$$

Program załącza tylko plik nagłówkowy `iostream`.

Przykładowe wykonanie

```
 In: 1000
Out: 3.14059
```

2.2.15 Tri: Liczby trzycyfrowe - grupowo

Napisz program `tri` wypisujący na standardowe wyjście w kolejności rosnącej wszystkie liczby trzycyfrowe, których cyfra setek to 1, 2, 5, 6, 7, lub 9, cyfra dziesiątek jest potęgą dwójki, cyfra jedności jest parzysta, a suma cyfr dzieli się przez 7. Program wypisuje te liczby przy pomocy odpowiednich pętli i załącza tylko plik nagłówkowy `iostream`.

Wykonanie

```
Out: 124 142 214 248 284 518 520 588 610 626 644 680 716 786 914 948 984
```

2.2.16 Trigonometry: Tabele trygonometryczne

Napisz program `trigonometry`, który drukuje na standardowe wyjście w kolejnych kolumnach kąt w stopniach od 0° do 180° z krokiem 10° oraz sinus i kosinus tego kąta sformatowane jak poniżej. Program wypisuje wyniki przy pomocy odpowiednich pętli i załącza tylko pliki nagłówkowe `cmath`, `iomanip` oraz `iostream`.

Wykonanie

```
Out:   0  0.000  1.000
Out:  10  0.174  0.985
Out:  20  0.342  0.940
Out:  30  0.500  0.866
Out:  40  0.643  0.766
Out:  50  0.766  0.643
Out:  60  0.866  0.500
Out:  70  0.940  0.342
Out:  80  0.985  0.174
Out:  90  1.000  0.000
Out: 100  0.985 -0.174
Out: 110  0.940 -0.342
Out: 120  0.866 -0.500
Out: 130  0.766 -0.643
Out: 140  0.643 -0.766
Out: 150  0.500 -0.866
Out: 160  0.342 -0.940
Out: 170  0.174 -0.985
Out: 180  0.000 -1.000
```

2.2.17 Truth: Tabela prawdy

Napisz program `truth` wypisujący na standardowe wyjście tabelę prawdy operacji $(p \vee q) \wedge r$. Kolejne kolumny zawierają odpowiednio wartości p, q, r, oraz $(p \vee q) \wedge r$. Program wypisuje tabelę przy pomocy odpowiednich pętli i nie używa instrukcji wyboru, instrukcji warunkowej ani operatora warunkowego. Program załącza tylko pliki nagłówkowe `iomanip` oraz `iostream`.

Wykonanie

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