Laboratorium 2: „Pętle i instrukcje warunkowe”

dr inż. Leszek Ciopiński
dr inż. Arkadiusz Chrobot
dr inż. Grzegorz Łukawski

4 września 2026

1. Wprowadzenie

Pierwsza część instrukcji zawiera informacje o instrukcjach warunkowych i instrukcjach wielokrotnego wyboru. Druga część poświęcona jest pętlom i powiązanych z nimi instrukcjami w języku C, a trzecia zawiera informacje na temat wybranych metod numerycznych. GDB.

2. Instrukcje warunkowe

Do instrukcji warunkowych zaliczamy instrukcję warunkową `if`, operator trójargumentowy `?` oraz instrukcję wielokrotnego wyboru `switch`.

Przykład instrukcji warunkowej `if` jest podany w listingu 1.

```
if(a==b)
        a=5;
else
        b=5;
```

Listing 1: Przykład instrukcji `if`

Warunek w nawiasach okrągłych nie musi być warunkiem prostym. Można go zbudować przy użyciu dowolnych operatorów, choć najczęściej używa się operatorów logicznych i relacyjnych. **Uwaga**: język C dopuszcza użycie w warunku instrukcji przypisania `=`, która podobna jest w zapisie do operatora porównania `==`. Te operatory działają w odmienny sposób i dosyć często użycie przypisania zamiast porównania jest pomyłką programisty. Czasem jednak jest to efekt zamierzony, bo taki zapis może okazać się z pewnych względów wygodny. Kompilator GCC ostrzega przed takimi pomyłkami, jeśli użyta jest flaga `-Wall` podczas kompilacji.

Słowo kluczowe `else` i następującą po nim część instrukcji można pominąć. Warto zauważyć, że w języku C instrukcja znajdująca się przed tym słowem kluczowym musi kończyć się średnikiem. Po warunku lub słowie `else` zamiast pojedynczej instrukcji może występować cały blok instrukcji zaczynający się i kończący nawiasami klamrowymi. Niektóre osoby zalecają stosowanie nawiasów klamrowych nawet dla pojedynczych instrukcji, co w pewnych sytuacjach może ustrzec programistę przed trudnymi do wykrycia błędami.

Zamiast instrukcji warunkowej można użyć operatora trójargumentowego. Jeśli warunek stojący przed znakiem zapytania jest prawdziwy, to operator zwraca wartość wyrażenia stojącego za pytajnikiem, w przeciwny przypadku, wartość wyrażenia stojącego za dwukropkiem. Listing 2 zawiera instrukcję `if` i operator `?` użyte do tej samej operacji - wybrania większej z wartości.

```
int a=5, b=4, max;

if(a>b)
        max=a;
else
        max=b;

max=(a>b)?a:b;
```

Listing 2: Instrukcja `if` i operator trójargumentowy

Listing 3 zawiera przykład zapisu instrukcji wielokrotnego wyboru.

```c
int a;

switch(a) {
        case 1: puts("Jeden");
                break;
        case 2: puts("Dwa");
                break;
        case 3: puts("Trzy");
                break;
        default: puts("Inna wartość");
}
```

Listing 3: Instrukcja wielokrotnego wyboru

Słowo kluczowe `default` (będące odpowiednikiem słowa `else` w instrukcji `if`) i związany z nim wariant nie muszą występować. Proszę zauważyć, że każdy wariant (ang. *case*) kończy się instrukcją `break`. Jeśli jej nie umieścimy, to program będzie realizował instrukcje z następnych wariantów, aż do napotkania `brake` lub zakończenia instrukcji wyboru. Taki zapis nie musi być koniecznie błędem, może być celowym działaniem. Instrukcję `break` można zastąpić instrukcją `return`, jeśli chcemy zakończyć działanie funkcji lub programu, albo równoważną jej inną instrukcją.

3. Instrukcje iteracyjne

W języku C dostępne są trzy podstawowe rodzaje pętli[1]. Są to pętle `for`, `while` i `do…while`.

3.1. Pętle `for`

Schematycznie pętla `for` może być przedstawiona następująco:

```
for(inicjacja;warunek;krok) instrukcja;
```

,gdzie `inicjacja` oznacza nadanie wartości początkowej zmiennej licznikowej, `warunek` oznacza warunek, który musi być spełniony, aby pętla mogła się wykonywać, a krok oznacza instrukcję zmieniającą wartość licznika. Język C pozwala użyć w pętli `for` zmiennej typu `float` lub `double` w charakterze licznika. Można też użyć więcej niż jednego licznika pętli. Listing 4 zawiera przykłady użycia pętli `for`.

[1] Za czwarty rodzaj można uznać „pętle” skonstruowane z użyciem instrukcji `goto`, ale nie będą one opisywane w tej instrukcji.

```c
int a;
for(a=0;a<5;a++)  /* Ta pętla powtarza się 5 razy. Wbrew pozorom nie jest to
        ;            pętla pusta - po jej wykonaniu zmienna a ma wartość 5. */

for(a=1;a<=5;a++)              // Ta pętla też wykonuje się 5 razy, ale wartość
        printf("%d\n",a);      // końcowa licznika pętli jest równa 6. W pętli
                               // wywoływana jest również funkcja printf, która
                               // wypisuje na ekran wartość licznika pętli.

for(a=0;a<7;a+=2)              // Nowością w tej pętli jest zwiększanie jej
        printf("%d\n",a);      // licznika o 2 w każdej iteracji. Wartość
                               // końcowa zmiennej a wynosi 8 (nie wyświetli
                               // się na ekranie).

for(a=7;a>0;a--)               // W tej pętli wartość licznika jest zmniejszana
        printf("%d\n",a);      // o jeden przy każdym powtórzeniu.

int i,j;
for(i=7,j=0;i>j;j++,i--)       // W tej pętli użyto dwóch liczników, jeden
        printf("%d %d\n",i,j); // (zmienna i) jest zmniejszany, a drugi (zmienna j)
                               // jest zwiększany.

double x;
for(x=0.0;x<0.5;x+=0.01)       // W tej pętli użyto jako licznika zmiennej
        printf("%.10lf\n",x);  // typu double.

a=0;
for(i=0; i<5; i++) {           // Zazwyczaj w pętli wykonywany jest cały
        a+=i;                  // blok instrukcji, a nie pojedyncza
        printf("%d\n",a);      // instrukcja.
}
```

Listing 4: Pętle `for`

3.2. Pętle `while` i `do…while`

Pętla `while` powtarza się tak długo, jak długo spełniony jest warunek w niej zawarty. Jeśli ten warunek nie jest nigdy spełniony, to pętla nigdy się nie wykona. Pętla `do…while` wykona się zawsze przynajmniej raz, bowiem warunek jest sprawdzany po wykonaniu zawartych w niej instrukcji. Każda z tych pętli powinna zawierać wyrażenie, które po określonej liczbie iteracji powinno sprawić, że warunek wykonania pętli stanie się fałszywy. Listing 5 zawiera kilka przykładów takich pętli.

```
char a=0;
while(a!='q')                // Pętla wykonywana jest do momentu naciśnięcia
        scanf("%c",&a);      // przez użytkownika klawisza q, a potem klawisza
                             // Enter

int x=0,y=0;
while(y>=0) {                // Pętla działa do momentu wprowadzenia przez
        scanf("%d",&y);      // użytkownika liczby ujemnej. Wewnątrz pętli
        x+=y;                // wykonywane są dwie instrukcje.
}

while(getchar()!='q')         // Ta pętla jest równoważna pierwszej w działaniu.
        ;                     // Funkcja getchar() zwraca kod ASCII znaku odczytanego
                              // ze standardowego wejścia. Proszę zwrócić uwagę na
                              // średnik kończący pętlę.

do                            // Ta pętla jest równoważna w działaniu pierwszej
        scanf("%c",&a);       // i trzeciej pętli. Proszę zauważyć, że zmienna
while(a!='q');                // a nie jest inicjowana przed pętlą, bo zanim
                              // zostanie zbadany warunek, to jej wartość jest
                              // zmieniana.

x=y=0;
do {                          // W tej pętli wykonywane są dwie instrukcje.
        x+=1;
        y*=y;
} while(x!=10);
```

Listing 5: Pętle `while` i `do…while`

Warunki w obu typach pętli podlegają tym samym regułom, co warunki w instrukcji warunkowej `if`.

3.3. Instrukcje `break` i `continue`

Wewnątrz pętli, w instrukcji warunkowej `if` można użyć instrukcji `break`, która przerywa wykonanie pętli. Listing 6 zawiera odpowiedni przykład.

```
int a;
for(a=0;a<10;a++) {
        printf("%d\n",a);
        if(a==5)
                break;
}
```

Listing 6: Instrukcja `break`

Instrukcja `break` może być użyta także w pętlach `while` i `do…while`. Warto zauważyć, że jeśli umieścimy tę instrukcję wewnątrz instrukcji `switch`, to będzie ona kończyła działanie poszczególnych przypadków, a nie pętli (patrz listing 7).

```c
int i;

for(i=0;i<3;i++)
        switch(a) {
                case 0: puts("zero");
                        break;
                case 1: puts("jeden");
                        break;
                case 2: puts("dwa");
       }
```

Listing 7: Użycie instrukcji `switch` w pętli `for`

W podobny sposób wewnątrz pętli może być użyta instrukcja `continue`. Jej działanie jest jednak inne - przerywa ona wykonanie bieżącej iteracji pętli i rozpoczyna wykonanie następnej. Przykład pokazano w listingu 8.

```c
int i;
for(i=0;i<11;i++) {             // Pętla wypisze tylko liczby parzyste.
        if(i%2)
                continue;
        printf("%d\n",i);
}
```

Listing 8: Użycie instrukcji `continue` w pętli

Instrukcji `continue` można użyć również wewnątrz pozostałych pętli.

3.4. Pętle nieskończone

Listing 9 zawiera przykłady realizacji pętli nieskończonych w języku C. Takie konstrukcje też mogą być przydatne, ale w przeciwieństwie do tych zaprezentowanych niżej muszą posiadać jakieś instrukcje, które kończyłyby ich wykonanie, np. opisaną wcześniej instrukcję `break`.

```c
for(;;);   // Pusta, nieskończona pętla for.
while(1);  // Pusta, nieskończona pętla while.
do; while(1); // Pusta nieskończona pętla do...while.
```

Listing 9: Przykłady pętli nieskończonych

4. Metody numeryczne wyznaczania wartości

W niniejszym rozdziale zaprezentowane zostaną wybrane metody numeryczne służące do przybliżonego wyznaczania wartości pierwiastka funkcji lub obliczania wartości funkcji nieliniowej w dowolnym punkcie dziedziny.

4.1. Szereg MacLaurina

Szereg MacLaurina jest szczególnym przypadkiem szeregu Taylora. W matematyce może on być użyty do wyznaczania wartości funkcji takich jak $sin(x), cos(x), e^x$.

Dla funkcji $sin(x)$ szereg ten ma postać :

$$\sin(x) = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} \quad = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots \tag{1}$$

,gdzie x jest miarą kąta wyrażoną w radianach.

Z punktu widzenia programisty warto zauważyć, że każdy kolejny wyraz tego szeregu różni się od poprzedniego o czynnik $(-1) \cdot \frac{x^2}{(2 \cdot i) \cdot (2 \cdot i + 1)}$, gdzie $i = 1, 2, \ldots$ jest numerem pozycji wyrazu w szeregu (jedynka przypada dla $\frac{x^3}{3!}$).

4.2. Szereg Leibniza

Jednym ze sposobów obliczenia wartości liczby π jest zastosowanie wzoru na szereg Leibniza: [2]

$$\sum_{n=1}^{\infty} (-1)^{n-1} \frac{1}{2n-1} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots = \frac{\pi}{4} \tag{2}$$

Ta metoda wyznaczania wartości liczby π jest dosyć wolna i na wynik jej obliczeń trzeba czasem poczekać kilkadziesiąt sekund, nawet w przypadku nowoczesnych komputerów.

4.3. Metoda Newtona

Metoda Newtona jest metodą numerycznego wyznaczania pierwiastka funkcji. Może ona być użyta do wyznaczania wartości pierwiastka kwadratowego z dodatniej liczby rzeczywistej. Zakładając, że y, to liczba, której pierwiastka poszukujemy, a x to szacowana wartość tego pierwiastka, otrzymujemy wzór:

$$x_{k+1} = \frac{(x_k + \frac{y}{x_k})}{2} \tag{3}$$

,gdzie k jest numerem kolejnego przybliżenia wartości pierwiastka. Jako wartość początkową x można przyjąć $x_0 = 1$. Ponieważ pierwsza wartość uzyskana z metody może okazać się niezadowalająca, metodę tą należy używać w pętli. Należy jednak sprawdzać, czy dalsze jej używanie wpływa na poprawę wyników.

5. Zadania

1. Zastosuj kod z listingu 3 w programie i sprawdź, przy użyciu debuggera, jak on działa, wykonując go krokowo. Następnie usuń z instrukcji `switch` słowa kluczowe `break` i ponów to ćwiczenie. Wartości zmiennej `a` powinna być wprowadzana z klawiatury przy użyciu funkcji `scanf()`.

2. Napisz program, który będzie trzykrotnie wypisywał na ekranie liczby naturalne od 1 do 7. Każda seria liczb powinna być wypisana w osobnym wierszu i przy użyciu innego rodzaju pętli niż pozostałe (`while`, `do…while` lub `for`). Sprawdź, przy użyciu debuggera jak ten program działa, wykonując go krokowo.

3. Napisz program, który będzie odliczał słownie do startu rakiety, ale od liczby, którą poda użytkownik, np. użytkownik podaje 5, a program wypisuje w kolejnych wierszach: „pięć”, „cztery”, itd.

4. Napisz program, który przy użyciu dowolnej pętli, policzy sumę ciągu arytmetycznego. Wyraz początkowy, końcowy oraz różnicę wyrazów program powinien otrzymać od użytkownika. W kodzie programu nie wolno Ci zastosować wzoru na sumę ciągu arytmetycznego.

5. Powtórz zadanie 4 dla pozostałych pętli, które zostały opisane w instrukcji.

6. Napisz program, który przy użyciu dowolnej pętli, policzy sumę ciągu geometrycznego. Wyraz początkowy, końcowy oraz iloraz wyrazów program powinien otrzymać od użytkownika. W kodzie programu nie wolno Ci zastosować wzoru na sumę ciągu geometrycznego. Program powinien również sprawdzać poprawność wprowadzonych przez użytkownika danych.

[2] „Encyklopedia Szkolna. Matematyka“, WSiP, Warszawa 1990

7. Powtórz zadanie 6 dla pozostałych pętli, które zostały opisane w instrukcji.

8. Postaraj się zapisać kod liczący sumę szeregu w zadaniu 4 jak najkrócej.

9. Przy pomocy pętli `while` i `do…while` oraz instrukcji warunkowych napisz program obliczający wartość silni dla argumentu z zakresu [0, 10].

10. Korzystając ze wzoru Maclaurina, oblicz wartość funkcji $\sin(x)$ dla $x = \pi/3$. Program powinien prosić użytkownika o podanie liczby wyrazów szeregu użytych do obliczenia wartości funkcji.

11. Korzystając z Szeregu Leibniza, oblicz wartość liczby π.

12. Napisz program, który wczyta od użytkownika nieujemną liczbę rzeczywistą oraz liczbę powtórzeń, a następnie obliczy pierwiastek kwadratowy z tej liczby korzystając z metody Newtona.