Wykład 4

4. Instrukcje

4.1. Instrukcja wyrażeniowa

4.2. Instrukcja grupująca

4.3. Instrukcja warunkowa *if* z pojedyńczym wyborem

4.4. Instrukcja warunkowa *if-else* z możliwością wielu wyborów

4.5. Instrukcja wyboru *switch*

Instrukcje:

- umożliwiają zapis algorytmu,
- służą do sterowania przebiegiem programu (podejmowania decyzji)

Klasyfikacja:

- proste – obejmują jedną operację (zakończone średnikiem),
- złożone – wiele operacji zgrupowanych w nawiasach { }.

Podstawowe instrukcje:

- wyrażeniowa,
- grupująca,
- instrukcja warunkowa,
- instrukcja wyboru,
- instrukcja iteracyjna (pętla).

4.1. Instrukcja wyrażeniowa

```
Wyrażenie;
```

Instrukcja wyrażeniowa to wyrażenie zakończone średnikiem. Wykonanie instrukcji wyrażeniowej polega na opracowaniu wyrażenia i odrzuceniu jego wyniku.

Np.
```
a = 5;
a++;
b = 7;
c = a+b;
;               // instrukcja pusta
```

W szczególnym przypadku, gdy przed średnikiem nie ma żadnego wyrażenia instrukcja wyrażeniowa jest instrukcją pustą. Jest ona wykorzystywana jeśli w pewnych przypadkach komputer ma nic nie robić.

4.2. Instrukcja grupująca

```
{
    instrukcja_1;
    instrukcja_2;
        ....
    instrukcja_N;
}
```

Instrukcja grupująca jest instrukcją złożoną, która składa się z ciągu instrukcji zawartych w nawiasach klamrowych { }. Jest ona traktowana jak pojedyncza instrukcja wyrażeniowa.

Jeśli w instrukcji grupującej występują definicje lub deklaracje to taka instrukcja jest *blokiem*.
Np.
```
{
    int k, j, b=2;
    k++;
    j = k + b;
}
```

4.3. Instrukcja warunkowa *if*

Umożliwia podjęcie decyzji w przypadku, gdy jest spełniony określony warunek. Jest ona często nazywana *instrukcją rozgałęzienia* ponieważ w wyniku jej wykonania program podejmuje decyzję, którą z dwóch ścieżek należy wybrać.

if (wyrażenie) *instrukcja*;

Jeśli wyrażenie jest prawdziwe (wartość różna od zera – może być dodatnia lub ujemna) to wykonywana jest *instrukcja*.

Np. Jeśli –5 <= x <= 10, to zwiększ licznik.

```
int licznik = 0;
if (-5 <= x && x <= 10) licznik++;

a = -1;
if (a) licznik++;  // -1 jest różne od 0; zwiększany jest licznik

char z = '\0';      // zmienna z ma wartość 0
if (z) licznik++;   // instrukcja nie zostanie wykonana !

a=1; b=3; c=7;
if (a>0 &&  b==5 || c==7) printf("Licznik = %i", ++licznik);
// najpierw a> && b==5 potem || c==7; jest 0 || 1 = 1; wydruk licznik;
```

Wyrażenia logiczne wykonywane od lewej do prawej. Priorytet && większy niż priorytet ||.

Instrukcja realizowana w przypadku prawdziwości wyrażenia może być prosta lub złożona.

```
if (wyrażenie)
{
    instrukcja_1;      // instrukcja zakończona średnikiem
    instrukcja_2;
        ...
    instrukcja_N;
}
```

Jeśli wartość wyrażenia jest różna od zera, to zostaną wykonane wszystkie instrukcje zawarte w nawiasach klamrowych. Jeśli wyrażenie ma wartość równą 0, to żadna z instrukcji nie zostanie wykonana.

Np.
```
if (a!=5)
{
    if (b<10) c+ = a-b;
    a = 2*b;
}
```

Jeśli wartość zmiennej a jest różna od 5, to instrukcje zostaną wykonane. Jeśli a ma wartość 5, to żadna z instrukcji nie zostanie wykonana.

```
if (a>0)
    if (b<-5 || b>7)
        if  (d) b=d+10;
```

Jeśli a>0 i b$\in (-\infty,-5) \cup (7, \infty)$ i d jest różne od 0, to b = d + 10.

```
if (a ==b) c = d;   // jeśli a równe b, to przypisz do c wartość d

if (a!=0) c++;      // jeśli a różne od 0, to zwiększ c = c+1

if (a) b+=c;        // jeśli a nie jest zerem, to b = b + c

if (!a) b-=c;       // jeśli a jest zerem, to b = b-c ( lub if (a==0) b-=c;)
```

Przykład 4.1. Wyznaczanie pierwiastków rzeczywistych równania kwadratowego $Ax^2 + Bx + C = 0$ (wykorzystanie if).

```
#include <iostream.h>
#include <conio.h>
#include <stdio.h>
#include <math.h>
#include <process.h>

double a,b,c;
double delta;
double x1, x2;

void main(void)
{
 clrscr();
```

```
 cout << "\nPodaj a : ";   cin >> a;
 cout << "\nPodaj b : ";   cin >> b;
 cout << "\nPodaj c : ";   cin >> c;

if (a==0)
{                              // a = 0
  cout << "\na = 0";
  if (b==0)
  {                            // b = 0
    cout << "\nb = 0";
    if (c==0)
      { cout << "\nRownanie spelnione dla kazdego x";
        exit(0);
      }
      if (c!=0)
       { cout << "\nBrak rozwiazan";
         exit(0);
       }
  }
    x1 = -c/b;        // a=0 , b!=0
    cout << "\nPierwiastek x1 = " << x1;
}
  if (a!=0)
  {
      delta = b*b - 4*a*c;
      if (delta < 0 ) cout << "\nPierwiastki zespolone";

      if (delta >= 0)
     {
        x1 = (-b-sqrt(delta))/(2*a);
        x2 = (-b+sqrt(delta))/(2*a);
        if (delta ==0)  cout << "\nx1 = " << x1;
        if (delta >0) {    cout << "\nx1 = " << x1;
                           cout << "\nx2 = " << x2;
                      }
     }
  }
   getch();
}
```

Przykład 4.2. Dane są liczby rzeczywiste A, B, C, D. Wyprowadzić na ekran nazwy równych sobie zmiennych.

```
#include <iostream.h>
#include <conio.h>
#include <stdio.h>
#include <math.h>
#include <process.h>

double a,b,c,d;

void main(void)
{
 clrscr();
 cout << "\nPodaj a : ";  cin >> a;
 cout << "\nPodaj b : ";  cin >> b;
 cout << "\nPodaj c : ";  cin >> c;
 cout << "\nPodaj d : ";  cin >> d;

 if (a==b && a==c && a==d) { cout << "\na=b=c=d"; goto end; }

 if (a==b && b==c && a!=d) { cout << "\na=b=c"; goto end; }
 if (a==b && a!=c && a==d) { cout << "\na=b=d"; goto end; }
 if (a!=b && a==c && a==d) { cout << "\na=c=d"; goto end; }
 if (a!=b && b==c && c==d) { cout << "\nb=c=d"; goto end; }

 if (a==b && c==d) { cout << "\na=b & c=d"; goto end; }
 if (a==c && b==d) { cout << "\na=c & b=d"; goto end; }
 if (a==d && b==c) { cout << "\na=d & b=c"; goto end; }

 if (a==b) { cout << "\na=b"; goto end; }
 if (a==c) { cout << "\na=c"; goto end; }
 if (a==d) { cout << "\na=d"; goto end; }
 if (c==d) { cout << "\nc=d"; goto end; }
 if (b==d) { cout << "\nb=d"; goto end; }
 if (b==c) { cout << "\nb=c"; goto end; }

end:
   getch();
}
```

4.4. Instrukcja warunkowa *if-else*

Instrukcja *if* umożliwia wybór pomiędzy wykonaniem a niewykonaniem instrukcji. Natomiast instrukcja *if-else* umożliwia wybór pomiędzy dwoma czynnościami w zależności od wartości wyrażenia.

if (wyrażenie) *instrukcja_1*; // średnik po instrukcji
else
instrukcja_2;

Jeśli wartość wyrażenia jest różna od zera, to wykonana zostanie *instrukcja_1*. W przeciwnym wypadku *instrukcja_1* zostanie opuszczona i wykonana zostanie *instrukcja_2*.

Np. int a=1, b=2, c=0;

```
        if (a+b > 3) a++;
              else c = a + b;
```

Ponieważ a+b = 3 wykonana zostanie druga instrukcja, tj. c = a + b.

Za pomocą instrukcji *if-else* można decydować o wykonaniu lub niewykonaniu grupy instrukcji.

```
if (wyrażenie)
{
        <grupa_instrukcji_1>;
}
        else
{
        <grupa_instrukcji_2>;
}

Np.   if (a==b)  c = d;
                 else  c = b;
        if (a < 10)
        {       if (b <7) c+ = a + b;
                     else b++;
        }
                else  {  c = 0;
                         a+= b;
                      }
```

Przykład 4.3. Wyznaczanie pierwiastków równania kwadratowego $Ax^2 + Bx + C = 0$ (wykorzystanie if-else).

```
#include <iostream.h>
#include <conio.h>
#include <stdio.h>
#include <math.h>
#include <process.h>

double a,b,c;  double delta;  double x1, x2, re, im;

void main(void)
{
 clrscr();
 cout << "\nPodaj a : ";  cin >> a;
 cout << "\nPodaj b : ";  cin >> b;
 cout << "\nPodaj c : ";  cin >> c;

if (a==0)
{
  cout << "\na = 0";
  if (b==0)
  {
    cout << "\nb = 0";
    if (c==0)
      cout << "\nRownanie spelnione dla kazdego x";
     else
           cout << "\nBrak rozwiazan";
     exit(0);
  }
   x1 = -c/b;     cout << "\nPierwiastek x1 = " << x1;
}
  else
  {
    delta = b*b - 4*a*c;
    if (delta >= 0)
    {  x1 = (-b-sqrt(delta))/(2*a);
       x2 = (-b+sqrt(delta))/(2*a);
       if (delta ==0)  cout << "\nx1 = " << x1;
                 else {    cout << "\nx1 = " << x1;
                           cout << "\nx2 = " << x2;
                      }
    }
```

```
    else       // pierwiastki zespolone
    {
       im = sqrt(-delta)/(2*a);
       re = -b/(2*a);
       cout << "\nz1 = " << re << " + i*(" << im << ")";
       cout << "\nz2 = " << re << " - i*(" << im << ")";
    }
 }
  getch();
}
```

Instrukcja *if-else* może być wielokrotnie zagnieżdżona.

```
if  (warunek_1)
{       <instrukcje_1>;          }

else  if (warunek_2)
       {       <instrukcje_2>;   }

    else      if  (warunek_3)
              {          <instrukcje_3>;   }

...

         else  if (warunek_N)
                  {     <instrukcje_N>;  }

            else  <instrukcje_N+1>;
                     // wykonywane jeśli wszystkie warunki if (...) są fałszywe.
```

Jeśli warunek_1 jest spełniony, to wykona się grupa instrukcji_1, a pozostałe sprawdzenia if zostaną opuszczone. W przeciwnym przypadku instrukcje_1 zostaną pominięte i nastąpi sprawdzenie wyrażenia warunek_2. W podobny sposób analizowane są pozostałe warunki.

W rozpatrywanym przypadku są wykonywane tylko instrukcje związane z prawdziwym warunkiem, a pozostałe są pomijane. Jeżeli żaden warunek nie jest prawdziwy, to wykonywane są instrukcje związane z ostatnim else.

```
Np.   if (delta > 0.0) { cout << "\nDwa pierwiastki rzeczywiste";  }
      else
           if (delta == 0.0)  { cout << "\nDwa identyczne pierwiastki";  }
                else { cout << "\nDwa pierwiastki zespolone";  }
```

Jeśli delta >0.0, to wyprowadzona zostanie informacja, że równanie ma dwa pierwiastki, a pozostałe informacje zostaną pominięte.

```
int a=3, b=3, c=0;

if (a+b > 5)
       if (b>=3)  { c++; }          // wykona się c++; c =1
             else  {  b = a; c--; }
c += a+b;                           // wykona się c = c + a+b; c=1+3+3 = 7
```

Jeśli a+b > 5, to sprawdzany jest warunek b>=3. Jeśli warunek a+b > 5 nie jest spełniony, to wykonywana jest instrukcja c += a+b;

Warunek *else* dotyczy najbliższej instrukcji if, która nie posiada *else*.

```
int a=2, b=3, c=0;

if (a+b > 5)
       if (b>=3) { c++; }
             else  {  b = a; c--; }
else c = a+b;                                   // wykona się c = 2+3 = 5
```

Można również wykorzystać nawiasy do pokazania związków pomiędzy if oraz else.

```
int a=2; b=3; c=0;

 if (a+b > 5)
 {
       if (b>=3) c++;
             else  {  b = a; c--; }
       c = a+b;
 }
  else
     if (c>0) { c = c + a;  b++; }
        else {  b = b + a; a++; }         // b = 5;  a=3;  c=0;
```

Przykład 4.4. Wyznaczanie minimum z trzech liczb (if-else).

```
#include <stdio.h>
#include <conio.h>

void main(void)
{
int A=-22;   int B=-50;   int C=-20;  int min;

clrscr();

  if (A<B)
      if (A<C) min=A;
      else min=C;
  else
      if (B<C) min=B;
      else min=C;

  printf("\nNajmniejsza liczba: %d", min);
  getch();
  }
```

Przykład 4.5. Wyznaczanie minimum z trzech liczb (if, &&).

```
#include <stdio.h>
#include <conio.h>

void main(void)
{
  int A=-22;    int B=-50;    int C=-57;   int min;
  clrscr();

  if (A<=B && A<=C) min=A;
  if (B<=C && B<=A) min=B;
  if (C<=B && C<=A) min=C;

  printf("\nNajmniejsza liczba: %d", min);
  getch();
  }
```

Przykład 4.6. Prosty kalkulator (if-else).

```
#include <iostream.h>
#include <conio.h>
#include <stdio.h>
#include <math.h>
#include <process.h>

char oper;
double a, b, wynik;

void main(void)
{
 clrscr();
 cout << "\nPodaj pierwszy argument : ";   cin >> a;
 cout << "\nWybierz operacje [+, -, /, *]: ";  cin >> oper;
 cout << "\nPodaj drugi argument : ";         cin >> b;
 cout << endl;

 if (oper == '+')
       wynik = a + b;

 else  if (oper == '-')
       wynik = a - b;

      else  if (oper == '*')
       wynik = a * b;

         else  if (oper == '/')
                {    if (b != 0.0) wynik = a/b;
                      else  { cout << "\nDzielenie przez zero"; exit(0); }
                }
                   else
                     { cout << "\nWybrano zla operacje"; exit(0); }

 cout << "\nWynik: ";
 cout << a << ' ' << oper << ' ' << b << " = " << wynik << endl;

 getch();
}
```

4.5. Instrukcja wyboru *switch*

Jeśli należy dokonać wyboru spośród wielu wariantów, to można wykorzystać instrukcję warunkową if-else. Jednak w większości przypadków wygodniej posłużyć się instrukcją wyboru wielowariantowego, która ma następującą postać:

switch (*wyrażenie_całkowite*) { **instr;** }

- *wyrażenie_całkowite* musi być typu całkowitego (np. char, int, long); wyrażenie_całkowite może być stałą, zmienną, wyrażeniem lub wywołaniem funkcji.

Instrukcja **instr;** może składać się z szeregu etykiet postaci:

case etykieta : <instrukcje>; break; oraz
default : <instrukcje>;

- Etykiety po słowie case mogą być *stałymi całkowitymi lub wyrażeniami zawierającymi wyłącznie stałe całkowite*.
- Etykiety stanowią wartości wyrażenia (*wyrażenie_całkowite*), które są analizowane w ramach instrukcji case.
- *Etykiety nie mogą być zmiennymi.* Do jednego przypadku w bloku *switch* może prowadzić kilka etykiet.
- Etykiety case są sprawdzane w pierwszej kolejności. Jeśli żadna z etykiet nie pasuje do wartości wyrażenia, to następuje przejście do wykonania instrukcji oznaczonych jako *default*.
- Instrukcja *break* kończy analizę przypadku *case* i powoduje opuszczenie bloku instrukcji wyboru.

```
switch (wyrażenie) {

case  stala_1 :  <instrukcje>;  break;
case  stala_2 : <instrukcje>;   break;
...
case stala_i: case stala_j: case stala_k: <instrukcje>;   break;
...
case  stala_N : <instrukcje>;   break;

default:      <instrukcje>;      break;       // w tej linii break jest opcjonalne
}
```

Np.

```
int delta = -1;

 switch(delta>=0)
   {
    case 0: cout << "Pierwiastki zespolone";   break;
    case 1: if (delta == 0) cout << "Dwa identyczne pierwiastki";
              else cout << "Dwa różne pierwiastki";   break;
    }
```

W szczególności: char stopien = ‘4’;

```
switch (stopien) {

case ‘6’ :  cout << ”\nDoskonale”;                 break;
case ‘5’ :  cout << ”\nBardzo dobrze”;             break;
case ‘4’ :  cout << ”\nDobrze”;                    break;
case ‘3’ :  cout << ”\nSlabo”;                     break;
 default :    cout << ”\nPostaraj się lepiej”;     break;
}
```

Własności switch

- Po każdej etykiecie *case* powinno występować wyrażenie typu całkowitego, będące analizowaną wartością selektora switch( ).
- Po etykiecie *case* nie może występować zakres wartości. Każda wartość musi występować z oddzielną etykietą *case*.
- Po każdym zestawie instrukcji dla danego wariantu należy umieścić instrukcję *break*, kończącą instrukcję *switch*. Jeżeli się tego nie zrobi, to sprawdzane będą następne warianty case.
- Zestaw instrukcji występujący po każdej etykiecie *case* nie musi być zamknięty nawiasami klamrowymi.
- Wariant *default* jest wykonywany jeżeli nie został spełniony żaden z przypadków *case*. Położenie etykiety default w bloku switch może być dowolne (np. default przed case lub w środku; najczęściej jednak default umieszczane na końcu bloku switch).

Przykład 4.7. Prosty kalkulator (switch).

```
#include <iostream.h>
#include <conio.h>
#include <stdio.h>
#include <math.h>
#include <process.h>

void main(void)
{
 char oper;
 double a, b, wynik;

 clrscr();
 cout << "\nPodaj pierwszy argument : ";
 cin >> a;
 cout << "\nPodaj drugi argument : ";
 cin >> b;
 cout << endl;

 cout << "\nWybierz operacje [+, -, /, *]: ";

 switch (oper = getche()) {

 case '+': wynik = a + b; break;

 case '-': wynik = a - b; break;

 case '*': wynik = a * b; break;

 case '/': if (b != 0.0) wynik = a/b;
           else
           { cout << "\nDzielenie przez zero"; exit(0); }
           break;
 default:  { cout << "\nWybrano zla operacje"; exit(0); }
}   //switch

 cout << "\nWynik: ";
 cout << a << ' ' << oper << ' ' << b << " = " << wynik << endl;

   getch();
}
```

Przykład 4.8. Badanie przynależności miesiąca do odpowiedniego kwartału. Analiza zakresów wartości: if-else & switch.

Zbiory dyskretnych wartości można określić za pomocą etykiet wielokrotnych.

```
void main(void)
{
 int mies;
 clrscr();
 cout << "\nPodaj numer miesiaca : ";   cin >> mies;    cout << endl;
 cout << "\nMiesiac o numerze " << mies << " jest z kwartalu";

 if (1<=mies && mies < 4) cout << " pierwszego.\n";
 else
 if (4<=mies && mies < 7) cout << " drugiego.\n";
 else
 if (7<=mies && mies < 10) cout << " trzeciego.\n";
 else
 if (10<=mies && mies < 13) cout << " czwartego.\n";
 else  cout << " nieznanego.\n";

 cout << "\nTo samo za pomoca switch ";
 cout << "\nMiesiac o numerze " << mies << " jest z kwartalu";

 switch (mies) {
 case 1:
 case 2:
 case 3: cout << " pierwszego.\n"; break;
 case 4:
 case 5:
 case 6: cout << " drugiego.\n"; break;
 case 7:
 case 8:
 case 9: cout << " trzeciego.\n"; break;
 case 10:
 case 11:
 case 12: cout << " czwartego.\n"; break;
 default:  { cout << " nieznanego.\n"; }
}   //switch
                getch();
}
```

Przykład 4.9. Wybór wariantu z wykorzystaniem enum.

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <time.h>

typedef enum  Kolory { R, G, B };

Kolory k_figura;   // zmienna przechowująca nr koloru

void main()
{
//   clrscr();
     randomize();   // inicjacja generatora
     int c = random(16);
     k_figura = (Kolory) (c % 4);  // losowy kolor figury

  switch (k_figura) {

    default: textcolor(LIGHTGRAY); cprintf("Kolor szary \n\r"); break;

    case R: textcolor(RED); cprintf("Kolor czerwony \n\r"); break;

    case G: textcolor(GREEN); cprintf("Kolor zielony \n\r"); break;

    case B: textcolor(BLUE);  cprintf("Kolor niebieski \n\r"); break;
   }
      getch();
}
```

Przykład 4.10. Wprowadzić z klawiatury rok, miesiąc i dzień. Jeżeli zapis jest poprawny, to wyprowadzić na ekran datę (miesiąc słownie).

```
#include <stdio.h>
#include <conio.h>

void main()
{
  int r,m,d;
  int maxd;     // liczba dni miesiąca
  clrscr();
  printf("Podaj rok: ");                    scanf("%i", &r);
  printf("Podaj miesiac [1..12]: ");   scanf("%i", &m);

  if  (r<0 || m<1 || m>12) { printf("Błędny zakres lat lub miesięcy\n"); }
  else
  {
  switch (m)
    {
     case 1: case 3: case 5: case 7: case 8: case 10: case 12:
                                              maxd=31; break;
     case 4: case 6: case 9: case 11: maxd=30; break;
     case 2: if ( (r % 4) || ( r % 100 == 0 && r % 400) ) maxd=28;
               else maxd=29; break;   // warunek roku zwykłego
               // lata zwykłe: 1700, 1800, 2100
               // lata przestępne: 4, 1600, 1996, 2000, 2400
    }
    printf("Podaj dzień [1..%i]: ",maxd);
    scanf("%i", &d);
  if (d<1 || d>maxd) { printf("Błędny zakres dni\n"); }
  else
  {
    printf("\n\n%i ", d);
    switch (m) {
    case 1: printf("%s ", "Stycznia"); break;
    case 2: printf("%s ", "Lutego"); break;
    case 3: printf("%s ", "Marca"); break;
    case 4: printf("%s ", "Kwietnia"); break;
    case 5: printf("%s ", "Maja"); break;
    case 6: printf("%s ", "Czerwca"); break;
    case 7: printf("%s ", "Lipca"); break;
    case 8: printf("%s ", "Sierpnia"); break;
    case 9: printf("%s ", "Września"); break;
    case 10: printf("%s ", "Października"); break;
    case 11: printf("%s ", "Listopada"); break;
    case 12: printf("%s ", "Grudnia"); break;
                }
   printf("%i roku",r);
   }
  }
   getch();
}
```