Dokumentacja programu Azdlzad1.cpp

1. Temat i analiza tematu

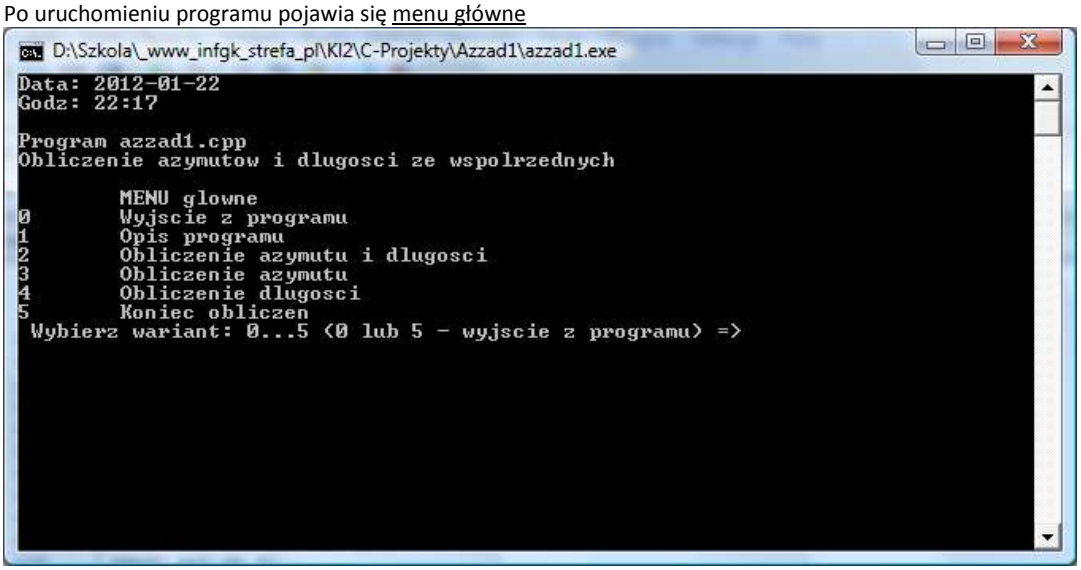
Program jest szkolną wersją prostego programu do obliczenia azymutu i długości ze współrzędnych.
Podaje się z klawiatury współrzędne x, y dwóch punktów, na podstawie których liczone są azymuty lub długości albo obie wartości równocześnie.

2. Algorytm ogólny

Azymuty obliczamy ze wzoru: Az[rad]=atan(Dy/Dx) z uwzględnieniem wartości Dy i Dx – ćwiartki układu współrzędnych oraz wartość Dx=0. Można też wykorzystać funkcję języka C
W programie utworzono własne funkcje

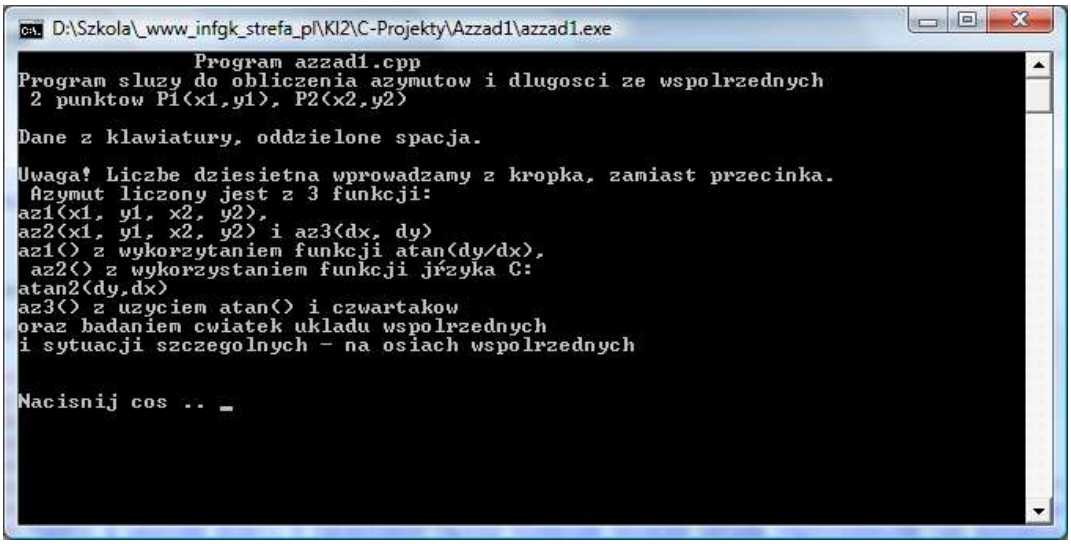
3. Specyfikacja użytkownika.

Program uruchamia się przez naciśnięcie nazwy programu lub wpisanie nazwy programu w okienku poleceń – najlepiej być w katalogu programu by nie podawać pełnej ścieżki dostępu.
Po uruchomieniu programu pojawia się menu główne

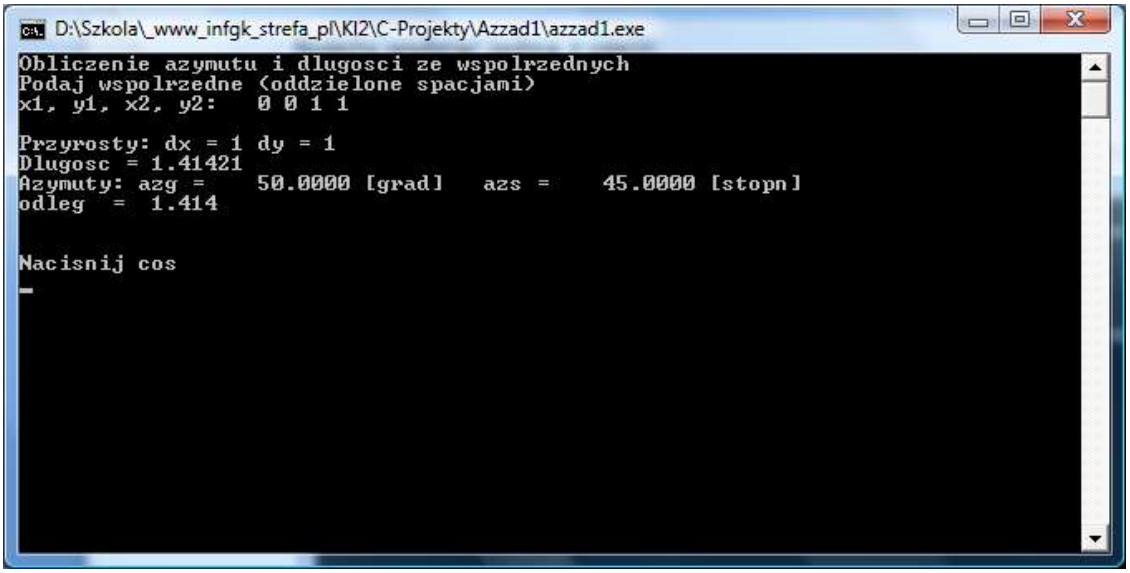


Należy wybrać jedną z opcji:

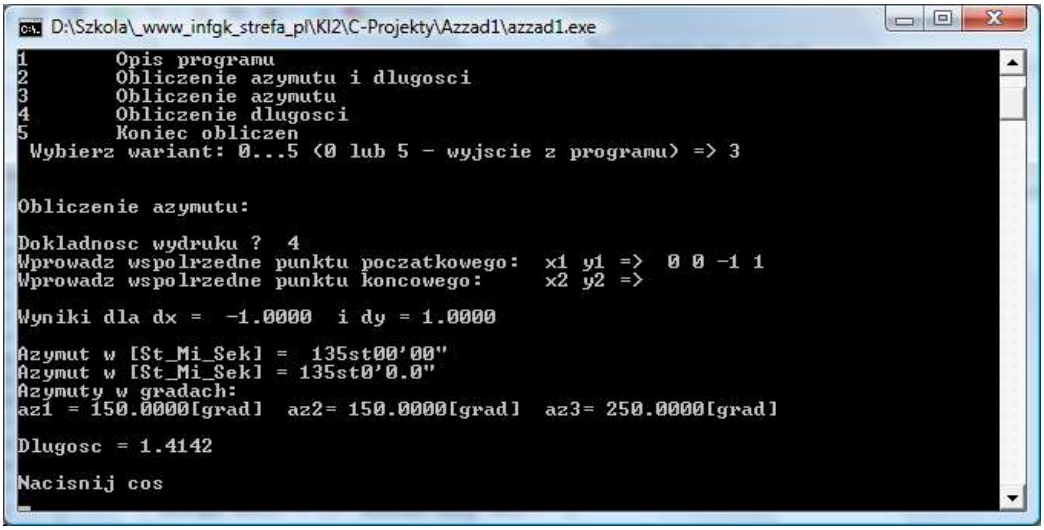
Po wybraniu **opcji 1** – krótki opis programu:



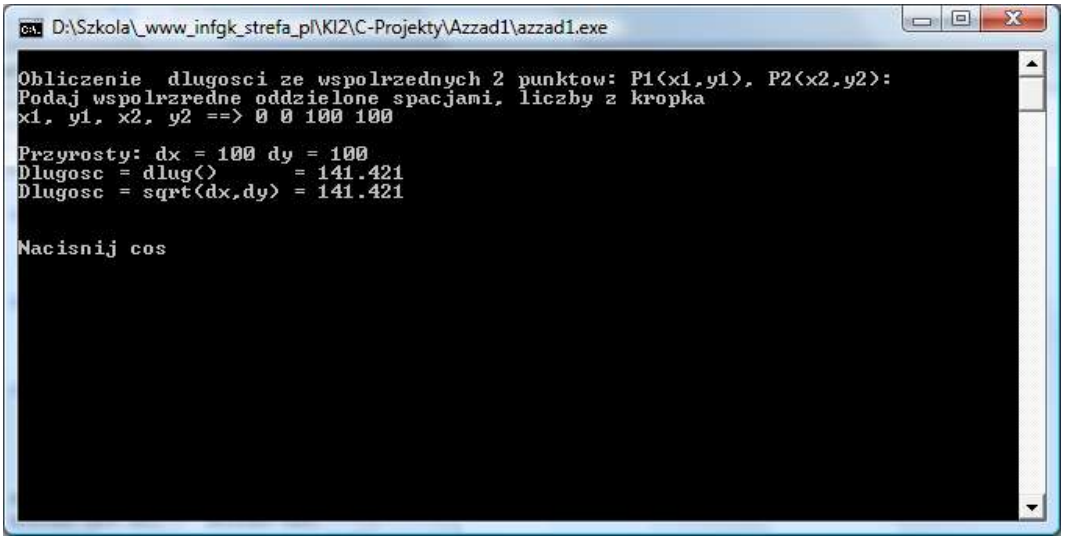
Opcja 2 – azymut i długość



Opcja 3: - obliczenie azymutu



Opcja 4 – obliczenie długości



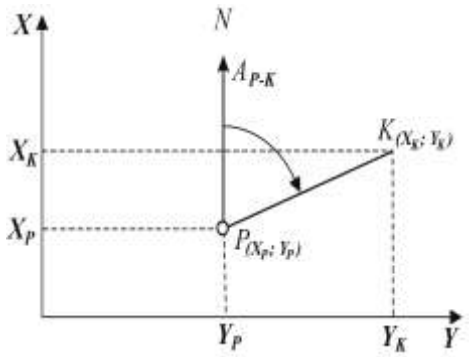
4. Specyfikacja wewnętrzna

4.1 Specyfikacja problemu algorytmicznego i opis użytych zmiennych

Problem algorytmiczny:
obliczenia azymutu wektora (odcinka skierowanego) o danych współrzędnych początku i końca.
Dane wejściowe:
współrzędne punktu początkowego P: XP, YP oraz punktu końcowego K: XK, YK.
Współrzędne te są oznaczane w programie odpowiednio przez x1, y1 oraz x2, y2.
Dane wyjściowe (wyniki):
azymut wektora PK: APK i długość odcinka PK: dPK.
Wyniki te są oznaczane różnie w poszczególnych funkcjach programu, zwykle z oznaczeniem początkowym a dla azymutu i d dla długości (np. a – azymut w mierze łukowej, azg – azymut w gradach, azs – w stopniach)

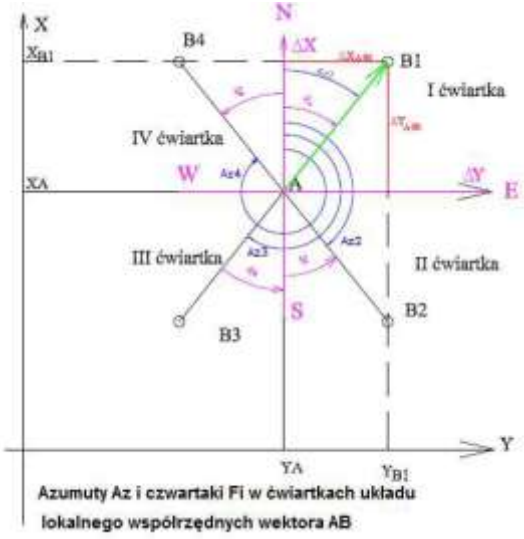
4.2 Algorytm szczegółowy

Pojęcie azymutu w geodezji
W geodezji obowiązuje prostokątny, prawoskrętny układ współrzędnych, w którym oś X skierowana jest pionowo do góry (kierunek północy) a oś Y jest skierowana poziomo w prawo - układ geodezyjny.
Kąty są liczone w prawo. W układzie matematycznym oś X jest pozioma, Y pionowa, kąty liczone są w lewo.
Wzory matematyczne są zgodne z matematycznymi.



Dowolny punkt P w tym układzie określamy parą współrzędnych X_P; Y_P , a punkt K parą X_K, Y_K
Azymutem lub kątem kierunkowym odcinka skierowanego (wektora) PK (o początku P i końcu K)
nazywamy kąt zawarty pomiędzy kierunkiem dodatniego kierunku osi równoległej do osi X w punkcie początkowym P a kierunkiem do drugiego punktu końcowego (K),
liczony zgodnie z ruchem wskazówek zegara i mogący przyjmować wartości od 0^g do 400^g, lub 0^o – 360^o lub 0 – 2PI [rad].

Azymut w układach ćwiartek układu współrzędnych, zaczepionego w punkcie początkowym A wektora AB



Algorytm obliczania azymutów

I metoda obliczenia azymutu

wykorzystanie funkcji arctan(DY/DX) i badanie położenia wektora względem układu lokalnego współrzędnych DX, DY, zaczepionego w punkcie P, równoległego do układu współrzędnych XY

1. Przyrosty współrzędnych odcinka PK:

$$\boxed{\Delta X_{PK} = X_K - X_P} \qquad \boxed{\Delta Y_{PK} = Y_K - Y_P}$$

2. Uwzględnienie znaków przyrostów współrzędnych

jeżeli $\Delta X_{PK} > 0$ oraz $\Delta Y_{PK} > 0$ to $A_{PK} = \operatorname{arctg} \frac{\Delta Y_{PK}}{\Delta X_{PK}}$

jeżeli $\Delta X_{PK} > 0$ oraz $\Delta Y_{PK} < 0$ to $A_{PK} = \operatorname{arctg} \frac{\Delta Y_{PK}}{\Delta X_{PK}} + 400^g$

jeżeli $\Delta X_{PK} < 0$ to $A_{PK} = \operatorname{arctg} \frac{\Delta Y_{PK}}{\Delta X_{PK}} + 200^g$

3. Przypadki szczególne, gdy kierunek wektora PK jest **równoległy do osi układu współrzędnych**

a) jeżeli $\Delta X_{PK} = 0$ oraz $\begin{cases} \Delta Y_{PK} > 0 \text{ to } A_{PK} = 100^g \\ \Delta Y_{PK} < 0 \text{ to } A_{PK} = 300^g \end{cases}$

b) jeżeli $\Delta Y_{PK} = 0$ oraz $\begin{cases} \Delta X_{PK} > 0 \text{ to } A_{PK} = 0^g \\ \Delta X_{PK} < 0 \text{ to } {}_{PK} = 200^g \end{cases}$

Azymut odcinka (wektora) KP = $A_{PK} + 200^g$

Zapis algorytmów obliczenia azymutu w I metodzie
– wykorzystanie funkcji **tan(dy/ dx)**

Zapis algorytmu funkcji obliczenia azymutu w gradach w postaci listy kroków

// np. funkcji az1(x1, y1, x2, y2)

1. Wczytaj współrzędne wektora PK: P(x1, y1), K(x2, y2)
2. Podstaw za pi wartość 4*atan(1) , za ro[grad]=rg wartość 200/pi, za ro[stopn] = RS wartość 180/pi
3. Wczytaj współrzędne początku x1, y1 oraz końca x2, y2
4. Oblicz przyrosty dx=x2-x1 i Dy=y2-y1
5. Jeśli dx jest równe 0 to: jeśli dy > 0 to za a podstaw pi/2 a w przeciwnym przypadku a = 1.5*pi, idź do kroku 6
6. Jeśli dx <>0 to za a podstaw atan(dy/dx)
7. Jeśli dx < 0 to a = a+pi, w przeciwnym przypadku (czyli dx=>0), jeśli dy <0 to a = a +2*pi
8. Obliczenie azymutu w gradach: azg=a*rg // w stopniach: azs=a*rs
9. Zwróć wartość azymutu (ewentualnie wyświetl wcześniej wyniki)

Pseudokod funkcji obliczenia azymutu w wersji angielskiej (if- jeśli, else – w przeciwnym razie

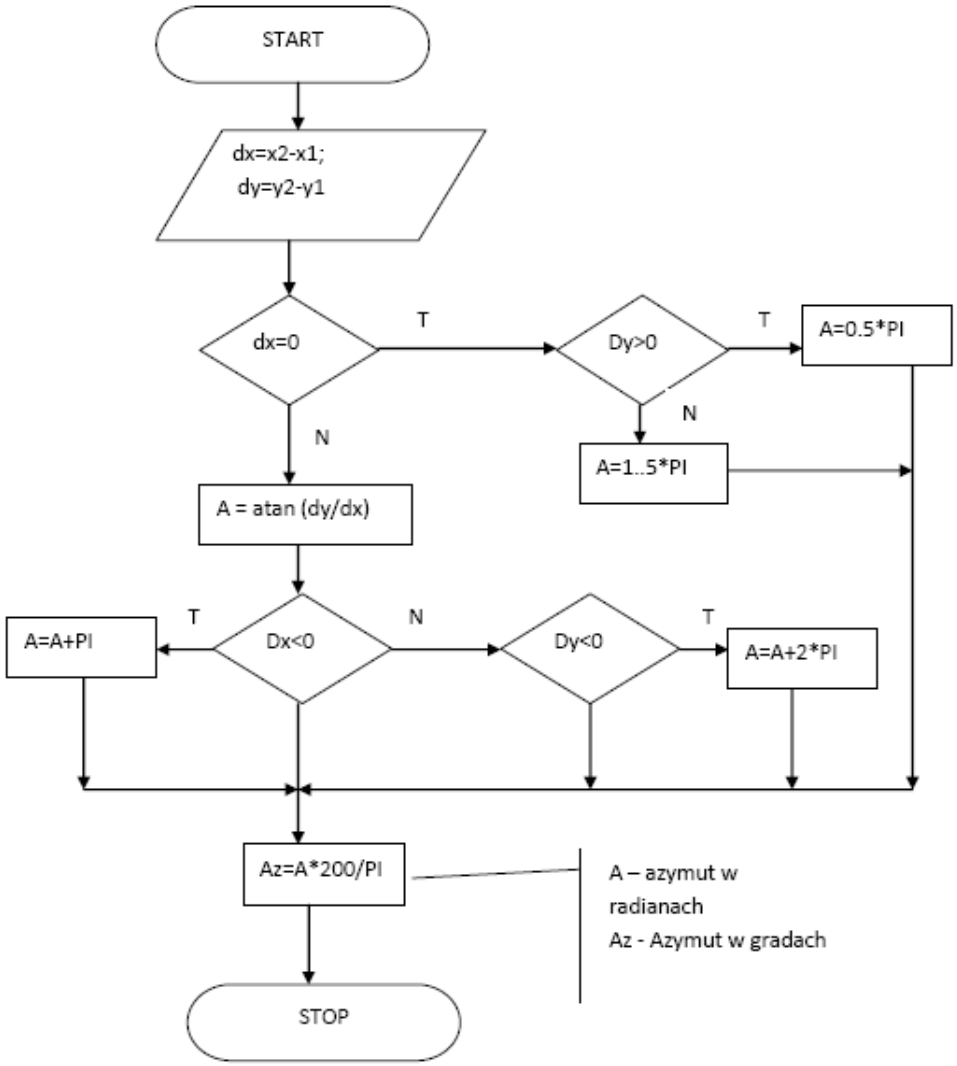
```
{
Pi=4*arctg(1);  RG=200/pi; RS=180/pi                    // Pi, RO[grad], RO[stop]
dx=x2-x1; dy=y2-y1  // przyrosty
if  dx=0  then {if dy>0 then a=pi/2 else a=1.5*pi;}     // azymut w radianach dla dx=0
else
{ if dx <0 then a=a+pi else a=a+2*pi}                   // azymut w radianach dla sx<>0
az=a*rg;
return az              // zwraca  wartość azymutu w gradach
}
```

Funkcja w języku C może być zapisana np. następująco

```
// azymut z wykorzystaniem atan()
double az1(double x1, double y1, double x2, double y2)
{
 double pi, rg,  dx, dy, d, a, az ;
 pi = 4.0 * atan(1.0);
 rg = 200.0 / pi;
 dx=x2-x1;  dy=y2-y1;

 if (dx==0)
 { if (dy>0) a=pi/2;   else  a=1.5*pi; }  // if dx==0
 else
 { // if (dx != 0)
  a = atan(dy / dx);
  if (dx < 0)  a = a + pi;
   else    {  if (dy < 0)  a = a + 2*pi;  } // if dy<0
 } // if (dx != 0)

 az = a * rg;  // obliczenie azymutu w gradach
 return az;
}
```

SCHEMAT BLOKOWY OBLICZENIA AZYMUTU

na podstawie danych współrzędnych 2 punktów : początku wektora P(x1, y1) i końca K(x2, y2)



II Metoda obliczenia azymutu – wykorzystanie funkcji atan2(dy, dx) języka C i M_PI = PI

200/M_PI – RO[gradowe]

$\boxed{\Delta X_{PK}}$ =dx=x2-x1; $\boxed{\Delta Y_{PK}}$ =dy=y2-y1;

Pseudokod zapisu funkcji az2(x1, y1, x2, y2) – lista kroków

Wczytaj x1, y1, x2, y2

Oblicz dx=x2-x1; dy=y2-y1

Azg=atan2(dy/dx)*200/M_Pi // M_PI = PI w bibliotece math.h języka C

Zwróć wartość Azg // azymut w radianach

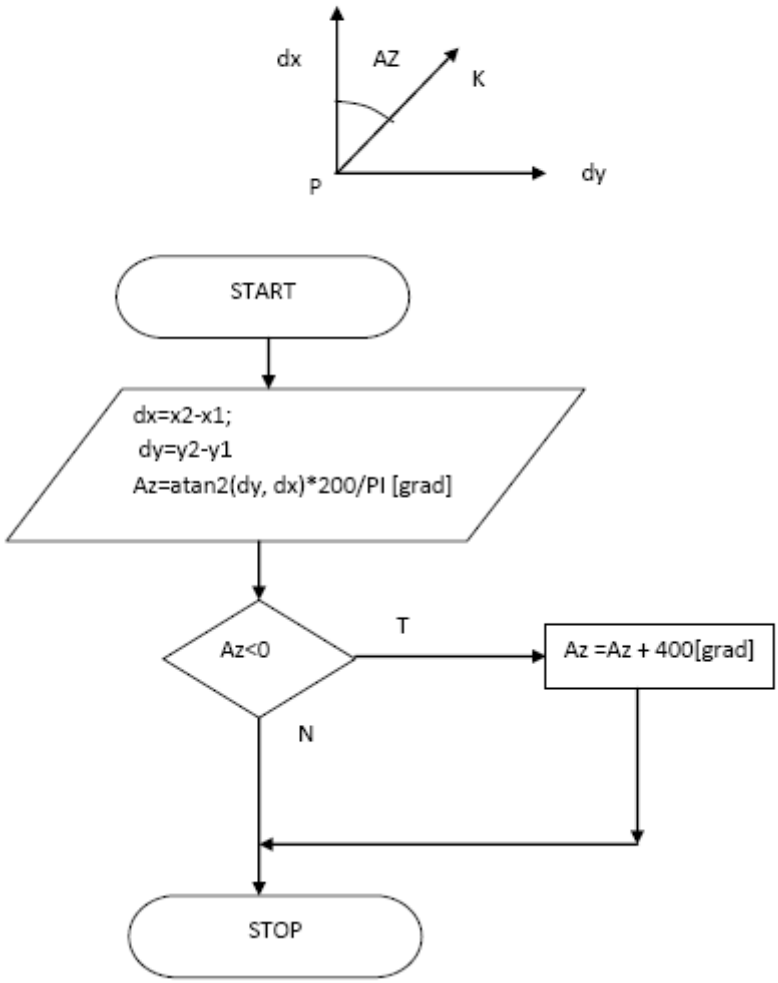
Funkcja w C/C++

```
// azymut z wykorzystaniem atan2(dy,dx)
double az2(double x1, double y1, double x2, double y2)
{
 double dx, dy, azg;
 dx=x2-x1; dy=y2-y1;
 azg = atan2(dy,dx) * 200.0/M_PI;
 if (azg <0) azg+=400.0;
 return azg;
}
```

SCHEMAT BLOKOWY OBLICZENIA AZYMUTU

na podstawie danych współrzędnych 2 punktów:
początku wektora P(x1, y1) i końca K(x2, y2)

Metoda z wykorzystaniem funkcji atan2(dy,dx)



III Metoda – tradycyjne wykorzystanie czwartaków

RG=200.0/M_PI; → Ro[grad]

Jeśli dx <> 0 to czw=RG*atan(dy/dx); *// czwartak w gradach*

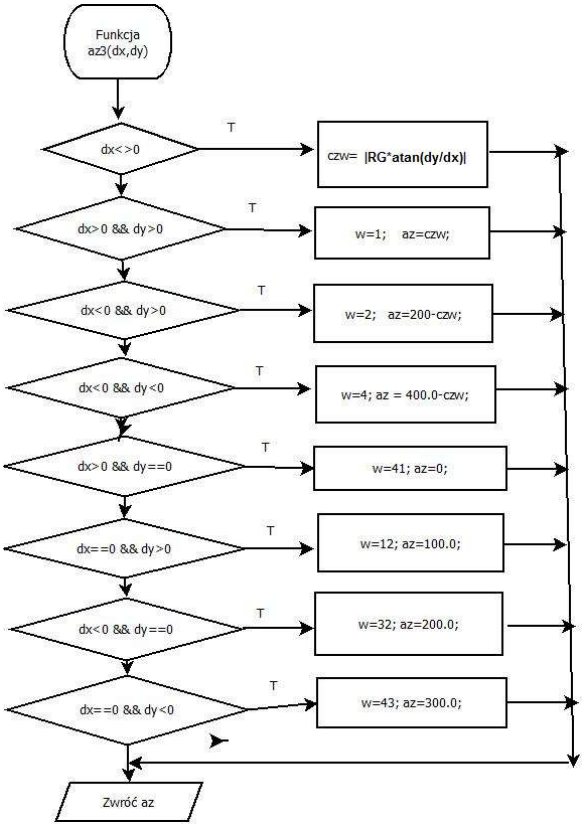
Uwzględnienie ćwiartek układu współrzędnych – w – ćwiartka (1 … 4)

w=41 – granica ćwiartki 4 i 1; w=12 – granica ćwiartki 1 i 2 itd.

Zapis w języku C np. postaci:

```
if (dx>0 && dy>0) {w=1; az = czw; break; } // I ćwiartka
if (dx<0 && dy>0) {w=2; az = 200.0-czw;  break; }
if (dx<0 && dy<0) {w=3; az = 200.0+czw; break; }
if (dx>0 && dy<0) {w=4; az = 400.0-czw; break; } // IV ćwiartka
if (dx>0 && dy==0) {w=41; az=0; break; } // granica ćwiartek 4 i 1
if (dx==0 && dy>0) {w=12; az=100.0; break; }
if (dx<0 && dy==0) {w=32; az=200.0; break; }
if (dx==0 && dy<0) { w=43; az=300.0; break; // granica ćwiartek 4 i 3
```

Schemat blokowy obliczenia azymutu na podstawie czwartaków



Pseudokod funkcji głównej

Wykonuj instrukcje
{
Wyświetl menu z opcjami programu
Wybierz wariant w (wczytaj wariant wyboru w) i uruchamiaj odpowiednią funkcję dopóki w różne od 0 i od 5.
przypadek 1: opis(); // wywołanie funkcji opis() i przerwanie przeszukiwania pętli, powrót na początek
przypadek 2: azymut_dlugosc(); przerwij;
// wywołanie funkcji azymut_dlugosc() i przerwanie przeszukiwania pętli, powrót na początek
przypadek '3': azymut(); przerwij; // funkcja azymut()
przypadek '4': dlugosc(); przerwij; // funkcja długość()
przypadek '0' oraz przypadek '5': koniec(); // przejdź do funkcji koniec() i zakończ program bo wybór 0 lub 5
inny: wyświetl napis "Niepoprawny wybor, powtorz";
} dopóki wariant wyboru różny od 5 i różny od 0
W zapisie języka C – pętla do while a w niej instrukcja wielokrotnego wyboru switch…case

```
do
 {
  clrscr(); // system("cls");
  cout << "Data: "; system("date/t");
  cout << "Godz: ";  system("time/t");
  cout << "\nProgram azzad1.cpp\n";
  cout << "Obliczenie azymutow i dlugosci ze wspolrzednych\n" << endl;
  cout << "      MENU glowne\n";
  cout << "0          Wyjscie z programu\n";
  cout << "1          Opis programu\n";
  cout << "2          Obliczenie azymutu i dlugosci\n";
  cout << "3          Obliczenie azymutu\n";
  cout << "4          Obliczenie dlugosci\n";
  cout << "5          Koniec obliczen\n";
  cout << " Wybierz wariant: 0...5 (0 lub 5 - wyjscie z programu) => ";
  cin >> w;
  switch(w)
  {
                    case '1': opis(); break;
```

```
                    case '2': azymut_dlugosc(); break;
                    case '3': azymut(); break;
                    case '4': dlugosc(); break;
                    case '0': case '5': koniec(); break;
                    default: cout << "Niepoprawny wybor, powtorz\n";
 } // switch

} while (w != '5' && w != '0');  // do
```

Specyfikacja programu

Program napisano w wersji języka C++ . Kompilowano programem Dev C++ oraz Code Blocks.

Program załącza **biblioteki** – pliki nagłówkowe dyrektywą #include:
#include <stdio.h>
#include <iostream>
#include <conio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <cstdlib>
Używa też polecenia using namespace std;

Zadeklarowano **stałe globalne**: const double RG=200.0/M_PI, RS=180.0/M_PI; - definicje RO – wartości radiana w gradach i stopniach.
Deklaracje funkcji umieszczono przed funkcją main() a definicje na końcu.

Funkcje:
void opis(); // opis programu
void czekaj(); // czeka na reakcję uzytkownika
void azymut_dlugosc(); // azymut i dlugosc
void azymut(); // azymut - wywoluje kolejne 3 funckcje: az1(), az2(), az3()
void dlugosc(); // oblicza dlugosc dwukrotnieL funkcja dlug() i standardowa sqrt()
void koniec(); // przed wyjsciem z programu
double dlug(double x1, double y1, double x2, double y2);
double az1(double x1, double y1, double x2, double y2);
double az2(double x1, double y1, double x2, double y2);
double az3(double dx, double dy);
char * zam_gr2st (double x); /* zamiana St na St-Mi_Sek */
char * grad2st(double grad);

Tabulogram programu

```
// Program azzad1.cpp - Obliczenie azymutu ze wspolrzednych
// Dyrektywy dolaczenie plikow naglowkowych
#include <stdio.h>
#include <iostream>
#include <conio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <cstdlib>
#define clrscr()  system("cls")
using namespace std;
// Stale: Ro[gradowe], Ro[stopniowe]  M_PI - Pi zdefiniowane w bibliotece math.h
const double RG=200.0/M_PI, RS=180.0/M_PI;

// Deklaracje funkcji
void opis(); // opis programu
void czekaj(); // czeka na reakcję uzytkownika
void azymut_dlugosc(); // azymut i dlugosc
void azymut(); // azymut - wywoluje kolejne 3 funckcje: az1(), az2(), az3()
void dlugosc(); // oblicza dlugosc dwukrotnieL funkcja dlug() i standardowa sqrt()
```

```
void koniec(); // przed wyjsciem z programu
double dlug(double x1, double y1, double x2, double y2);
double az1(double x1, double y1, double x2, double y2);
double az2(double x1, double y1, double x2, double y2);
double az3(double dx, double dy);
char * zam_gr2st (double x); /* zamiana St na St-Mi_Sek */
char * grad2st(double grad);

// Funkcja glowna
int main()
{
 char w;
 do
 {
  clrscr(); // system("cls");
  cout << "Data: "; system("date/t");
  cout << "Godz: ";   system("time/t");
  cout << "\nProgram azzad1.cpp\n";
  cout << "Obliczenie azymutow i dlugosci ze wspolrzednych\n" << endl;
  cout << "       MENU glowne\n";
  cout << "0          Wyjscie z programu\n";
  cout << "1          Opis programu\n";
  cout << "2          Obliczenie azymutu i dlugosci\n";
  cout << "3          Obliczenie azymutu\n";
  cout << "4          Obliczenie dlugosci\n";
  cout << "5          Koniec obliczen\n";
  cout << " Wybierz wariant: 0...5 (0 lub 5 - wyjscie z programu) => ";
  cin >> w;
  switch(w)
  {
                        case '1': opis(); break;
                        case '2': azymut_dlugosc(); break;
                        case '3': azymut(); break;
                        case '4': dlugosc(); break;
                        case '0': case '5': koniec(); break;
                        default: cout << "Niepoprawny wybor, powtorz\n";
  } // switch
 }while (w != '5' && w != '0');  // do
  return 0;
}

// Definicje funkcji uzytkownika

// funkcja opis() - procedura - nic nie zwraca
void opis()
{
 system("cls");
 cout << "                  Program azzad1.cpp\n";
 cout << "Program sluzy do obliczenia azymutow i dlugosci ze wspolrzednych\n";
 cout << " 2 punktow P1(x1,y1), P2(x2,y2)\n" ;
 cout << "\nDane z klawiatury, oddzielone spacja.  \n";
 cout << "\nUwaga! Liczbe dziesietna wprowadzamy z kropka, zamiast przecinka. \n ";
 cout << "Azymut liczony jest z 3 funkcji: \naz1(x1, y1, x2, y2),\naz2(x1, y1, x2, y2) i az3(dx, dy)\n";
 cout << "az1() z wykorzytaniem funkcji atan(dy/dx),\n az2() z wykorzystaniem funkcji języka C: \natan2(dy,dx)\n";
 cout << "az3() z uzyciem atan() i czwartakow \noraz badaniem cwiatek ukladu wspolrzednych \ni sytuacji szczegolnych - na
osiach wspolrzednych\n";
 cout << endl;
 cout << "\nNacisnij cos .. ";
 getch();
}

void koniec()
{  cout << "Koniec programu. Nacisnij Enter \n";  }

// procedura azymut_dlugosc() - funkcja nic nie zwraca
void azymut_dlugosc()
{
```

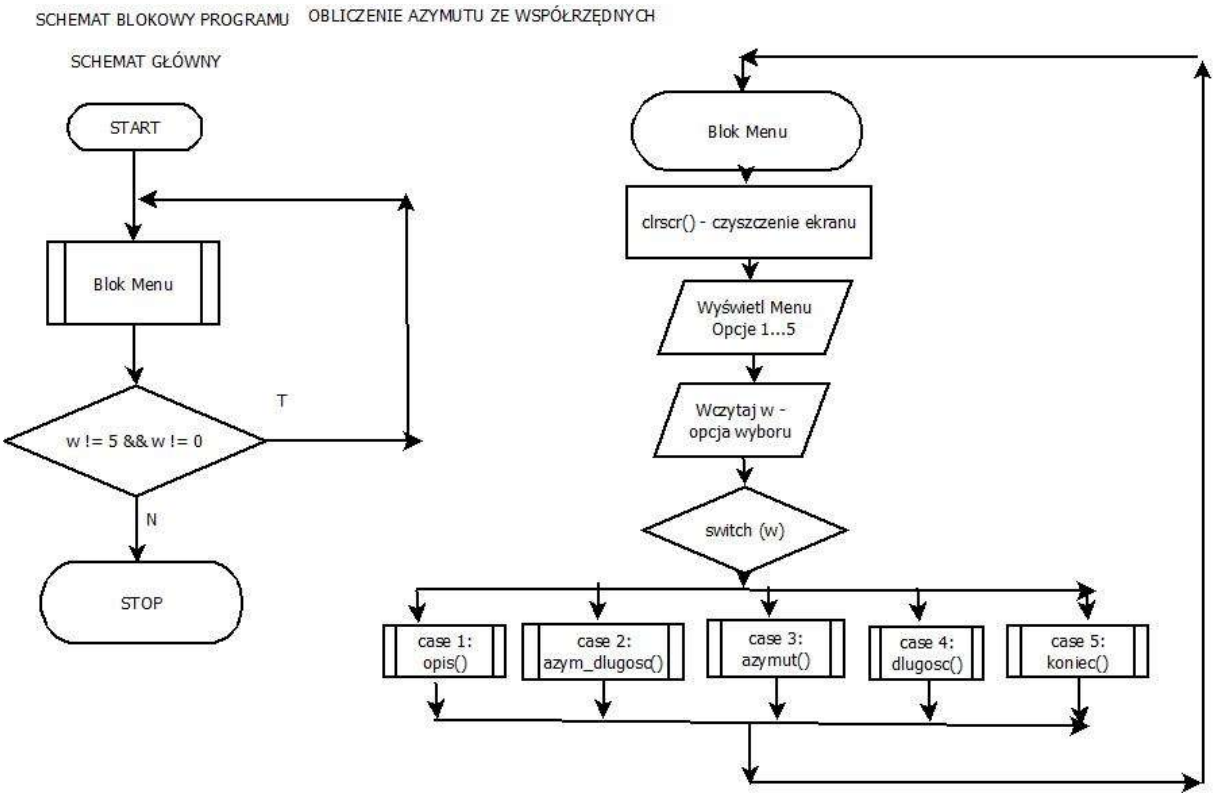
```
 double pi, rg, rs, x1, y1, x2, y2, dx, dy, d, a, azg, azs;
 pi = 4.0 * atan(1.0);  rg = 200.0 / pi;  rs = 180.0 / pi;
clrscr();
cout << "Obliczenie azymutu i dlugosci ze wspolrzednych\n";
cout << "Podaj wspolrzedne (oddzielone spacjami)\n";
cout << "x1, y1, x2, y2:   "; cin >> x1 >> y1 >> x2 >> y2;
dx=x2-x1; dy=y2-y1; d=sqrt(dx*dx+dy*dy); cout << endl;
cout << "Przyrosty: dx = " << dx << " dy = " << dy << endl;
cout << "Dlugosc = " << d << endl;
if (dx==0)
 {
  if (dy>0) a = pi/2;  else  a = 1.5*pi;
 } // if dx==0
else  // if (dx != 0)
{
 a = atan(dy/dx);
              if (dx < 0) //  dx  <0
                a = a+pi;
               else // dx >0
         { if (dy < 0)   a = a+2*pi;  }
}
azg = a * rg;  // obliczenie azymutu w gradach
azs = a * rs;  // obliczenie azymutu w stopniach
printf ("Azymuty: azg = %10.4f [grad]", azg);
printf ("    azs = %10.4f [stopn]\n", azs);
printf ("odleg  = %6.3f\n", dlug(x1, y1, x2, y2));
cout << endl; czekaj(); clrscr();
}

// procedura azymut - funkcja nic nie zwraca
void azymut()
{
double x1, y1, x2, y2, dx, dy, azm1, azm2, azm3, dl;
char *ksms, *kats; int d;
cout.setf(ios::fixed);  // zapewnia ze precision() odnosi sie do miejsc po kropce
cout.setf(ios::fixed);
cout << "\n\nObliczenie azymutu:\n";
cout << "\nDokladnosc wydruku ?  " ; cin >> d;
cout.precision(d); // precyzja (patrz wyzej o
cout << "Wprowadz wspolrzedne punktu poczatkowego:  x1 y1 =>  ";
cin >> x1 >> y1; cin.ignore();
cout << "Wprowadz wspolrzedne punktu koncowego:     x2 y2 =>  ";
cin >> x2 >> y2; cout << endl; cin.ignore();
dx=x2-x1; dy=y2-y1; azm1= az1(x1,y1, x2, y2);
azm2= az2(x1, y1, x2, y2); azm3= az3(dx,dy);
cout << "\nWyniki dla dx =  " << dx << "  i dy = " << dy << endl;
ksms = zam_gr2st(azm1); kats = grad2st(azm1); cout << endl;
cout << "Azymuty w gradach:\n";
cout << "az1 = " << azm1 << "[grad]  az2= " << azm2 << "[grad]  az3= " << azm3 << "[grad]" << endl;
dl=dlug(x1,y1, x2, y2);
cout << "\nDlugosc = " << dlug(x1,y1, x2, y2) << endl; czekaj();
}

// procedura (funkcja nic nie zwraca) na obliczenie dlugosci
void dlugosc()
{
double x1, y1, x2, y2, dx, dy, d;
system("cls");
cout << "\nObliczenie  dlugosci ze wspolrzednych 2 punktow: P1(x1,y1), P2(x2,y2):\n";
cout << "Podaj wspolrzredne oddzielone spacjami, liczby z kropka\n";
cout << "x1, y1, x2, y2 ==> "; cin >> x1 >> y1 >> x2 >> y2;
dx=x2-x1; dy=y2-y1; d=dlug(x1,y1,x2,y2); cout << endl;
cout << "Przyrosty: dx = " << dx << " dy = " << dy << endl;
cout << "Dlugosc = dlug()      = " << d << endl;
cout << "Dlugosc = sqrt(dx,dy) = " ;
printf("%.3lf\n",d); cout << endl; czekaj();
}
```

```
// czeka na nacisniecie klawisza
void czekaj()
 {
  cout << "\nNacisnij cos\n";    getch();    system("cls");
 }

// Funkcje dodatkowe - definicje
// azymut z wykorzystaniem atan()
double az1(double x1, double y1, double x2, double y2)
{
 double pi, rg,  dx, dy, d, a, az ;
 pi = 4.0 * atan(1.0);  rg = 200.0 / pi;
 dx=x2-x1;  dy=y2-y1;
 if (dx==0)
 { if (dy>0) a=pi/2;   else  a=1.5*pi; }  // if dx==0
 else
 { // if (dx != 0)
  a = atan(dy / dx);
  if (dx < 0)  a = a + pi;
   else    {  if (dy < 0)  a = a + 2*pi;  } // if dy<0
 } // if (dx != 0)
 az = a * rg;  // obliczenie azymutu w gradach
 return az;
}

// azymut z wykorzystaniem atan2(dy,dx)
double az2(double x1, double y1, double x2, double y2)
{
  double dx, dy, result;
  dx=x2-x1; dy=y2-y1;   result = atan2(dy,dx) * 200.0/M_PI;
  if (result <0) result+=400.0;   return result;
}

// azymut z wykorzystaniem czwartaka
double az3(double dx, double dy )
{
 unsigned w; double az=0, czw=0;
 if (dx !=0) czw=fabs(RG*atan(dy/dx));
 for (int i=1; i<8; i++)
 {
 if (dx>0 && dy>0) {w=1; az = czw; break; }
 if (dx<0 && dy>0) {w=2; az = 200.0-czw;  break; }
 if (dx<0 && dy<0) {w=3; az = 200.0+czw; break; }
 if (dx>0 && dy<0) {w=4; az = 400.0-czw; break; }
 if (dx>0 && dy==0) {w=41; az=0; break; }
 if (dx==0 && dy>0) {w=12; az=100.0; break; }
 if (dx<0 && dy==0) {w=32; az=200.0; break; }
 if (dx==0 && dy<0) { w=43; az=300.0; break; }
 }
 switch (w) {
        case 1 : az=czw;    break;
        case 2: az = 200.0-czw;  break;
        case 3: az = 200.0+czw; break;
        case 4: az = 400.0-czw; break;
        case 41: az = 0.0; break;
        case 12: az=100.0; break;
        case 32: az=200.0; break;
        case 43: 43; az=300.0; break;
        default : cout << endl << "Nieodpowiednie dane" << endl;  return(-1);
      }
 return  az;
}

/* Funkcja dlug */
double dlug(double x1, double y1, double x2, double y2)
{
```

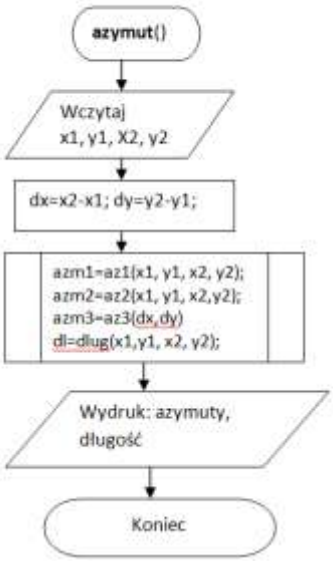
```
 double dx, dy, d;
 dx=x2-x1;  dy=y2-y1;  d=sqrt(dx*dx+dy*dy);
 return d;
 }

// Zamiana katow z gradow na stopnie, minuty i sekundy - 1 wersja
char * zam_gr2st (double x) /* zamiana St na St-Mi_Sek */
{
   char znak;    double   s, st, sr, m, mi, mr, se;
   double  xa, st1, min1;    long int  stc, mic, skc;
   char  stl[13], st1l[6], mil[3], skl[3], m1l[4], sk1l[4] ;
   char stp, mp, skp;    char kat[20];
   static char  * wyn1;
   xa=x;
   if (x<0)    {     xa=-x;     };
   s=xa*0.9;;
   sr = modf(s, &st);  // s - stopnie_dziesiet, st - stopnie calk, sr - reszta
   m=sr*60;    // minuty dziesietne
   mr=modf(m,&mi); // mi - minuty calk, mr - minuty reszta
   se=mr*60;    // sekundy dziesietne
   stc=st;      // stopnie calkowite
   mic=mi;      // minuty calkowite
   skc=(se+0.5);  // sekundy calkowite
   ltoa(stc, stl, 10); // long int na ASCII, system 10-ny
   ltoa(mic, mil, 10);
   ltoa(skc, skl, 10);

    if (mic<10) // jesli ilosc minut < 10
     { strcpy(m1l,"0");   strcat(m1l,mil); }
     else   strcpy(m1l,mil);

    if (skc<10)
      {  strcpy(sk1l,"0");  strcat(sk1l,skl); }
     else  strcpy(sk1l,skl);

    if (x<0)
      { strcpy(st1l,"-");  strcat(st1l,stl);  }
      else
      {  strcpy(st1l," ");  strcat(st1l,stl); };
    strcpy(kat,st1l); strcat(kat,"st");
    strcat(kat,m1l);  strcat(kat,"'");
    strcat(kat,sk1l); strcat(kat,"\"");
    cout << "\nAzymut w [St_Mi_Sek] = " << kat;
    wyn1=kat;
    return wyn1;
}

// Zamiana katow z gradow na stopnie, minuty i sekundy - 2 wersja
char * grad2st(double grad)
{
 long int stc, mic, sec, sk10;  long int sk;
 double std, mid, sed, skr;
 char  st_str[20];  char  mi_str[4];
 char  se_str[4];  char  se10_str[4];
 char kat_st[25];  static char  * wyn2;
 std=grad*0.9;  stc= floor(std);
 mid=(std-stc)*60.0;  mic=floor(mid);
 sed=(mid-mic)*60.0;  sec= floor(sed);
 skr=(sed-sec);  sk10=int(skr*10);
 stc=(int) stc;  mic=(int) mic;
 sec=(int) sec;  ltoa(stc, st_str, 10);
 ltoa(mic, mi_str, 10);  ltoa(sec, se_str, 10);
 ltoa(sk10, se10_str, 10);
 strcpy(kat_st,st_str); strcat(kat_st,"st");
 strcat(kat_st,mi_str);  strcat(kat_st,"'");
 strcat(kat_st,se_str); strcat(kat_st,".");
 strcat(kat_st,se10_str); strcat(kat_st,"\"");
```

```
 cout << "\nAzymut w [St_Mi_Sek] = " << kat_st;
 wyn2=kat_st;
 return wyn2;
}
```

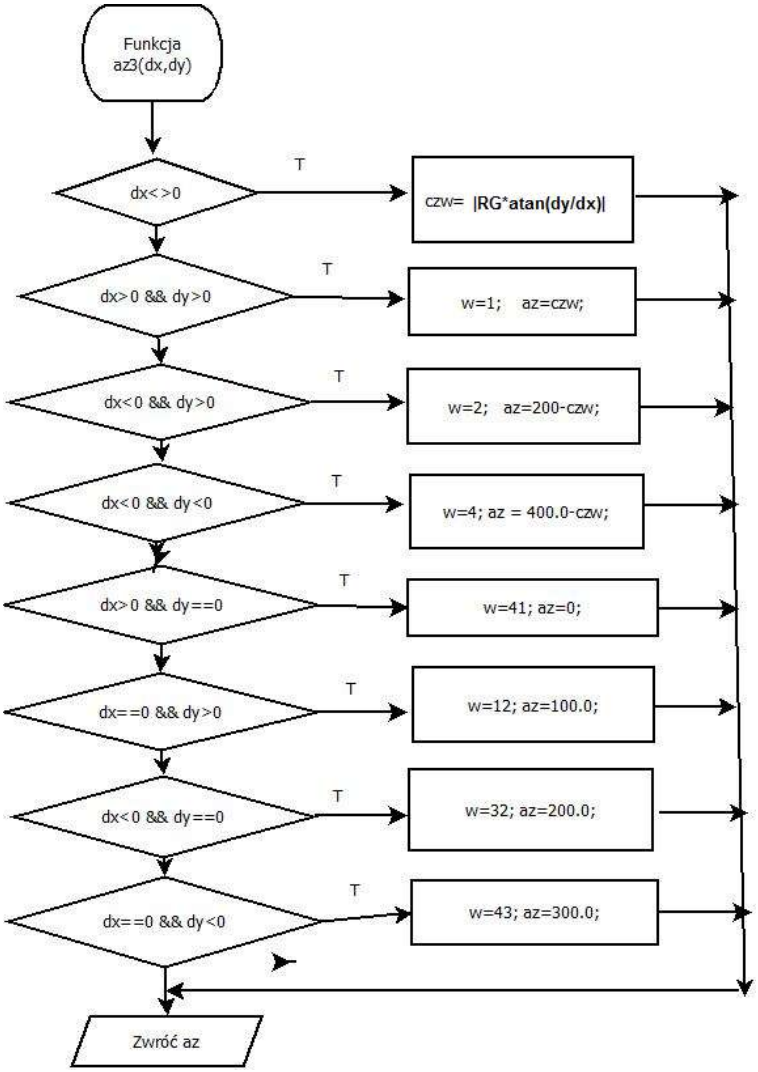
Schematy blokowe dotyczące niektórych funkcji programu



Funkcja void azymut() – procedura wczytania współrzędnych i obliczenia azymutu



Funkcja az3(dx, dy) – obliczenie azymutu z wykorzystaniem czwartaków.


Funkcja
az3(dx,dy)
dx<>0
T
czw= |RG*atan(dy/dx)|
dx>0 && dy>0
T
w=1; az=czw;
dx<0 && dy>0
T
w=2; az=200-czw;
dx<0 && dy<0
T
w=4; az = 400.0-czw;
dx>0 && dy==0
T
w=41; az=0;
dx==0 && dy>0
T
w=12; az=100.0;
dx<0 && dy==0
T
w=32; az=200.0;
dx==0 && dy<0
T
w=43; az=300.0;
Zwróć az