Część XV C++

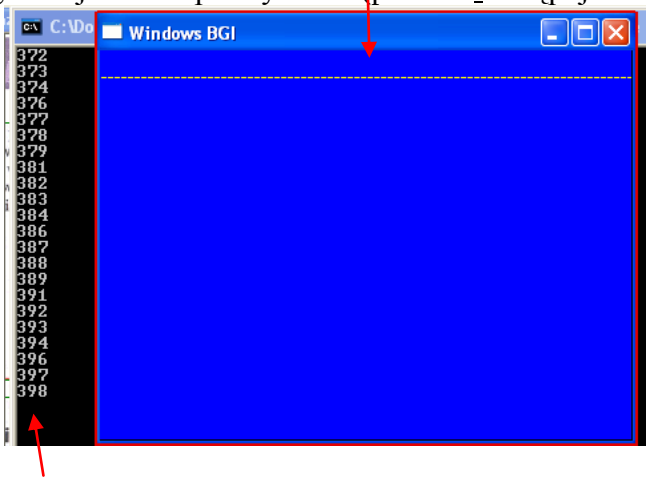
Instrukcja break przerywa działanie tylko tej pętli, w ciele której została wywołana.
Jeśli więc wywołamy break w pętli zagnieżdżonej w innej pętli, zostanie przerwane działanie tylko tej wewnętrznej pętli.

Następna iteracja - continue

Instrukcja **break** przerywa całkiem działanie pętli. Jednak w niektórych wypadkach potrzebne jest przerwanie tylko aktualnego przebiegu pętli, czyli przejście do następnej iteracji pętli. Do tego celu służy instrukcja continue. Jej działanie poznamy na przykładzie:

```
for (int x = 0; x < getmaxx(); x++)/*Tworzymy pętlę for, w której zmienna [x]
 w kolejnych przebiegach zmienia się o wartość 1 od 0 do maksymalnego rozmiaru okna.*/
{
      if (x%5==0) continue;/*W ciele funkcji umieszczamy instrukcję warunkową.
      Kiedy wartość zmiennej x jest podzieła przez 5 (dzielenie x przez 5 będzie
      bez reszty), zostaje wykonana instrukcja continue. Powoduje ona przerwanie
      aktualnego przebiegu pętli i rozpoczęcie wykonywania kolejnego (zmienna x jest
      zwiększana, zostaje sprawdzony warunek zakończenia pętli i instrukcje z ciała
      funkcji zostają ponownie wykonane).*/
      putpixel (x,20,YELLOW);
      cout << x << endl;
}
```

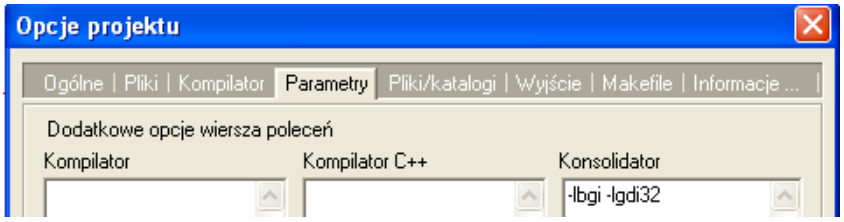
Gdy przebieg pętli nie zostaje przerwany, jest rysowany piksel w punkcie o współrzędnych [x],20. Efektem działania tego programu jest linia przerywana (przerwa następuje w co piątym punkcie linii).



Aby przekonać się, że przebieg pętli zostaje przerywany wtedy, gdy x jest podzielne przez 5, spójrzmy na wygenerowany dzięki instrukcji cout zapis -"brakuje" w nim liczb podzielnych przez 5.

Ćwiczenie 1

1. Utwórz nowy projekt w Dev C++ i zapisz go na pulpicie w folderze o nazwie **nazwisko52**
2. Z menu Projekt wybieramy Opcje projektu i w oknie klikamy na zakładkę Parametry W pole konsolidator wpisujemy: -lbgi -lgdi32 i zatwierdzamy Ok



3. Z menu Projekt wybieramy Dodaj do projektu,
4. Przechodzimy do Mój komputer →Dysk lokalny C → folder Dev-Cpp →folder include (katalogc:\dev-cpp\include) i dwukrotnie klikamy na ikonę winbgim 

5. Wprowadź do projektu modyfikacje tak aby wyglądał jak poniżej

```
#include <cstdlib>
#include <iostream>
#include <winbgim.h>

using namespace std;

int main(int argc, char *argv[])
{
    initwindow(400,300);
    setfillstyle(SOLID_FILL, BLUE);
    bar (0,0,getmaxx(),getmaxy());

    for (int x = 0; x < getmaxx(); x++)
    {
          if (x%5==0) continue;
          putpixel (x,20,YELLOW);
          cout << x << endl;
    }

    cout << endl << endl;
    while(!kbhit());
    closegraph();


    system("PAUSE");
    return EXIT_SUCCESS;
}
```

6. Z menu Uruchom wybieramy Kompiluj i uruchom
7. W oknie zapisz plik wskazujemy pulpit → folder nazwisko52 → otwieramy folder nazwisko52→ klikamy zapisz

Skocz do, czyli goto

W C++ występuje pewna instrukcja, której zdecydowanie nie zaleca się wykorzystywać. Jest to instrukcja goto pozwalająca na skok z jednego do innego miejsca programu. Dlaczego nie powinno się korzystać z instrukcji goto? Ponieważ są dużo lepsze sposoby rozwiązania problemów programistycznych niż goto - na przykład funkcje.

Skoro nie należy używać instrukcji goto, to po co o niej w ogóle pisać? Z dwóch powodów.
Po pierwsze dlatego, że warto wiedzieć, jak działa instrukcja goto - zawsze możemy się na nią natknąć w cudzych programach.
Po drugie dlatego, że może się zdarzyć, że zastosowanie goto okaże się lepszym rozwiązaniem niż każde inne. Poznamy teraz taką sytuację i na jej przykładzie zobaczymy, jak działa instrukcja goto

```
char znak1;
float r;
while(1) /* w nieskończonej pętli */
{
    cout << "Podaj promien kola: ";
    cin >> r; /* pobrana zostaje od użytkownika długośc promienia */
    while(1) /* zostaje uruchomiona druga nieskończona pętla */
    {
        cout << "1 - oblicz pole, 2 - oblicz obwod, " << endl;
        cout << "3 - podaj inny promien, q - zakoncz program" << endl;
        /* wybranie cyfry 3 powoduje wykonanie instrukcji break*/
        cin >> znak1;
        if (znak1=='1') cout << "Pole wynosi " << 3.14 * r * r << endl;
        else if (znak1=='2') cout << "Obwod wynosi " << 2 * 3.14 * r << endl;
        else if (znak1=='3') break;/* break powoduje wyjscie z aktualnej pętli
        i wyswietlenie ponownie prosby o podanie promienia */
        else if (znak1=='q') goto koniec; /* wybranie wczesniej q powoduje
        opuszczenie obydwu pętli jednoczesnie i wykonanie programu zostaje
        przeniesione do pierwszej instrukcji znajdujacej sie po etykiecie o nazwie koniec*/
    }
}

koniec:
cout << "Koniec programu!" << endl << endl; /* po opuszczeniu obu pętli zostaje wyswietlony
komunikat */
```

Ćwiczenie nr 2

1. Utwórz nowy projekt w Dev C++ i zapisz go na pulpicie w folderze o nazwie **nazwisko53**
2. Wprowadź do projektu modyfikacje tak aby wyglądał jak poniżej

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
    char znak1;
    float r;
    while(1)
    {
        cout << "Podaj promien kola: ";
        cin >> r;
        while(1)
        {
            cout << "1 - oblicz pole, 2 - oblicz obwod, " << endl;
            cout << "3 - podaj inny promien, q - zakoncz program" << endl;
            cin >> znak1;
            if (znak1=='1') cout << "Pole wynosi " << 3.14 * r * r << endl;
            else if (znak1=='2') cout << "Obwod wynosi " << 2 * 3.14 * r << endl;
            else if (znak1=='3') break;
            else if (znak1=='q') goto koniec;
        }
    }

    koniec:
    cout << "Koniec programu!" << endl << endl;

    system("PAUSE");
    return EXIT_SUCCESS;
}
```

3. Z menu Uruchom wybieramy Kompiluj i uruchom

W pętlach typu for wyrażenia inicjujące, warunkowe oraz zmieniające znajdują się w nagłówku pętli. Dzięki temu kod jest bardzo przejrzysty. W naszych przykładach pętli while i do... while w nagłówku znajduje się jedynie wyrażenie warunkowe,

```
int i = 0;
while(i<10){
    cout << i << " ";
      i++;
}
```

a zmienna iteracyjna zmieniana jest w ciele pętli. Jednak nie ma przeszkód, aby przy sprawdzaniu warunku przejścia do następnej iteracji pętli zmieniać jednocześnie wartość zmiennej iteracyjnej. Jak to robić? W wypadku najczęściej występującej zmiany - inkrementacji -możemy postąpić na przykład tak

```
int i = 0;
while(i++<10){
    cout << i << " ";

}
```

(dla pętli while) lub tak

```
int i = 0;
do {
     cout << i << " ";
}
while(i++ <9);
```

(w wypadku pętli do... while).

Taka pętla while działa następująco: na początku sprawdzany jest warunek **i** < 10 i niezależnie od wyniku tego sprawdzenia zmienna **i** zwiększana jest o 1. Jeżeli warunek jest prawdziwy, wykonywane są instrukcje z ciała pętli. Należy zwrócić uwagę, że warunek przed pierwszą iteracją sprawdzany jest przy wartości **i** wynoszącej 0, a instrukcje z ciała pętli są wykonywane przy **i** równym 1. Tak więc efektem działania takiej pętli jest wynik (inny niż w wypadku pętli ze zmianą **i** w ciele pętli).

W wypadku pętli do...while zmienna **i** przyjmuje wartości od 0 do 9.

Piszemy prawdziwy program

Dzięki wykorzystaniu instrukcji warunkowych, pętli programowych oraz instrukcji sterujących break i continue możemy z naszego prostego programu obliczającego pola różnych figur stworzyć funkcjonalną aplikację.

```
int main(int argc, char *argv[])
{
    char znak1, znak2;
    float a, b, pole;/*Na początku deklarujemy kilka niezbędnych zmiennych
    typu znakowego i zmiennoprzecinkowego*/
    while(1)/*Następnie tworzymy nieskończoną pętlę while- dzięki temu program
    będzie działał "w kółko" (później stworzymy oczy-wiście możliwość jego opuszczenia)*/
    {
        cout << "Wybierz figure: " << endl;
        cout << "1 - prostokat, 2 - kolo, 3 - trojkat, 0 - koniec " << endl;/*Wyświetlamy
        menu naszego programu i w zmiennej znaki zapisujemy wybór użytkownika*/
        cin >> znak1;

        switch (znak1){
            case '1': cout << "Podaj dlugosci bokow: ";
                      cin >> a >> b; pole = a * b; break;
            case '2': cout << "Podaj promien: ";
                      cin >> a; pole = 3.14 * a * a; break;
            case '3': cout << "Podaj dlugosc podstawy i wysokosc: ";
                      cin >> a >> b; pole = 0.5 * a * b; break;
                      /*Korzystając z instrukcji wyboru switch, w zależności od wybranej
                      figury pobieramy dane dotyczące figury oraz obliczamy pole wybranej
                      Figury */
            case '0': goto koniec;/*Gdy użytkownik zdecydował o zakończeniu programu
            (wybrał 0) , przenosimy działanie programu do etykiety [koniec:].
            Wtedy wyświetlony zostaje komunikat ,,Koniec programu'' i działanie programu
             się kończy */
            default: continue;/*Gdy użytkownik wybiera inny klawisz, za pomocą instrukgi
            [continue) przechodzimy do kolejnego przebiegu funkcji (czyli jeszcze raz
            wyświetlamy menu naszego programu).*/
        }
        cout << "Pole figury wynosi " << pole << endl;/*W zmiennej pole znajduje się obliczone
        przez nas pole figury. Wypisujemy więc tę wartość na ekranie */
        cout << "Wykreslic figure? (t/n): ";/* następnie zadajemy użytkownikowi pytanie, czy
        narysować figurę. */
        cin >> znak2;
        if (znak2=='t'){
            initwindow (400,400);
            /*Jeśli odpowiedź jest twierdząca, ponownie, wykorzystując instrukcję switch,
            wykreślamy odpowiednią figurę (wyświetlamy ją na środku okna o rozmiarach 400
            na 400 pikseli*/
            switch (znak1){
                case '1': bar(int(200-(a/2)),int(200-(b/2)),int(200+(a/2)),int(200+(b/2))); break;
                case '2': circle(200,200,(int)a); break;
                case '3': line(int(200-(a/2)),int(200-(b/2)),int(200+(a/2)),int(200-(b/2)));
                          line(int(200-(a/2)),int(200-(b/2)),int(200-(a/2)),int(200+(b/2)));
                          line(int(200-(a/2)),int(200+(b/2)),int(200+(a/2)),int(200-(b/2)));
                          break;

            }
            while(!getch());
            closegraph();
        }
    }
    koniec:
    cout << "Koniec programu!" << endl << endl;


    cout << endl << endl;
```

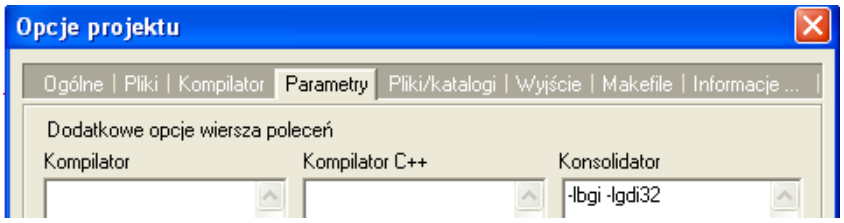
Etykieta jest po prostu specjalnie oznaczonym miejscem w kodzie. Aby wstawić etykietę, należy wpisać jej nazwę i dwukropek.

Funkcja **int** () zamienia liczbę znajdującą się w nawiasach na liczbę całkowitą.
Dzięki temu wartość zmiennej typu **float** możemy przypisać do zmiennej całkowitoliczbowej (oczywiście nie zostanie przypisana do niej część ułamkowa, lecz jedynie część całkowita).

Ćwiczenie 3

1. Utwórz nowy projekt w Dev C++ i zapisz go na pulpicie w folderze o nazwie **nazwisko54**

2. Z menu Projekt wybieramy Opcje projektu i w oknie klikamy na zakładkę Parametry W pole konsolidator wpisujemy: -lbgi -lgdi32 i zatwierdzamy Ok.



3. Z menu Projekt wybieramy Dodaj do projektu,

4. Przechodzimy do Mój komputer →Dysk lokalny C → folder Dev-Cpp →folder include

5. (katalogc:\dev-cpp\include) i dwukrotnie klikamy na ikonę winbgim 

6. Wprowadź do projektu modyfikacje tak aby wyglądał jak poniżej

```
#include <cstdlib>
#include <iostream>
#include <winbgim.h>

using namespace std;

int main(int argc, char *argv[])
{
    char znak1, znak2;
    float a, b, pole;
    while(1)
    {
        cout << "Wybierz figure: " << endl;
        cout << "1 - prostokat, 2 - kolo, 3 - trojkat, 0 - koniec " << endl;
        cin >> znak1;

        switch (znak1){
            case '1': cout << "Podaj dlugosci bokow: ";
                      cin >> a >> b; pole = a * b; break;
            case '2': cout << "Podaj promien: ";
                      cin >> a; pole = 3.14 * a * a; break;
            case '3': cout << "Podaj dlugosc podstawy i wysokosc: ";
                      cin >> a >> b; pole = 0.5 * a * b; break;
            case '0': goto koniec;
            default: continue;
        }
        cout << "Pole figury wynosi " << pole << endl;
        cout << "Wykreslic figure? (t/n): ";
        cin >> znak2;
        if (znak2=='t'){
            initwindow (400,400);
            switch (znak1){
                case '1': bar(int(200-(a/2)),int(200-(b/2)),int(200+(a/2)),int(200+(b/2))); break;
                case '2': circle(200,200,(int)a); break;
                case '3': line(int(200-(a/2)),int(200-(b/2)),int(200+(a/2)),int(200-(b/2)));
                          line(int(200-(a/2)),int(200-(b/2)),int(200-(a/2)),int(200+(b/2)));
                          line(int(200-(a/2)),int(200+(b/2)),int(200+(a/2)),int(200-(b/2)));
                          break;
            }
            while(!getch());
            closegraph();
        }
    }
    koniec:
    cout << "Koniec programu!" << endl << endl;


    cout << endl << endl;


    system("PAUSE");
    return EXIT_SUCCESS;
```

7. Z menu Uruchom wybieramy Kompiluj i uruchom
8. W oknie zapisz plik wskazujemy pulpit → folder nazwisko54 → otwieramy folder nazwisko54→ klikamy zapisz

Sprawdź czy program działa poprawnie, jeśli nie popraw błędy. Jeśli działa poprawnie wprowadź inne figury i obliczanie odwodów figur.

Wszystkie pliki z nazwiskiem i kolejnym numerem umieszczamy w swoim folderze nazwiskocplusplus na serwerze.