Część XIII C++

Czym jest pętla?

Pętla jest blokiem instrukcji, które wykonywane są w kółko (czyli po wykonaniu ostatniej instrukcji z bloku komputer wykonuje ponownie pierwszą instrukcję, później drugą i tak dalej). Oczywiście w większości wypadków takie ciągłe wykonywanie bloku instrukcji spowodowałoby zawieszenie się programu, więc pętlę należy w określonym momencie przerwać. Różnice między typami pętli występującymi w C++ polegają właśnie na sposobie przerywania ich działania.

Pętla typu while

Najprostszym typem pętli jest pętla typu while (z ang. dopóki). Jej ogólną postać można przedstawić tak:

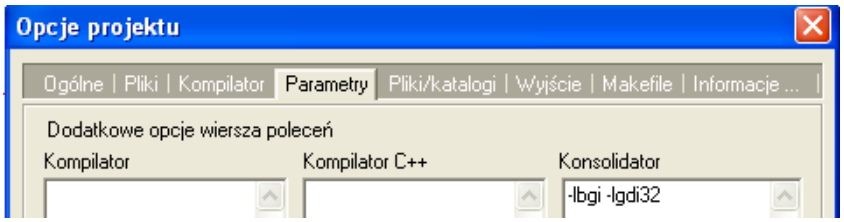
```
while (wyrażenie warunkowe)
{
 instrukcja 1;
 instrukcja 2;
 instrukcja N;
 }
```

Działanie pętli typu while możemy opisać w taki sposób: dopóki wyrażenie warunkowe jest prawdziwe, wykonuj instrukcje umieszczone w nawiasach klamrowych. Gdy warunek przestanie być spełniony (będzie fałszem), wykonywanie pętli się kończy i komputer przechodzi do wykonania następnej instrukcji (znajdującej się po pętli while).

Każde wykonanie bloku instrukcji nazywamy pojedynczym przebiegiem lub iteracją pętli.

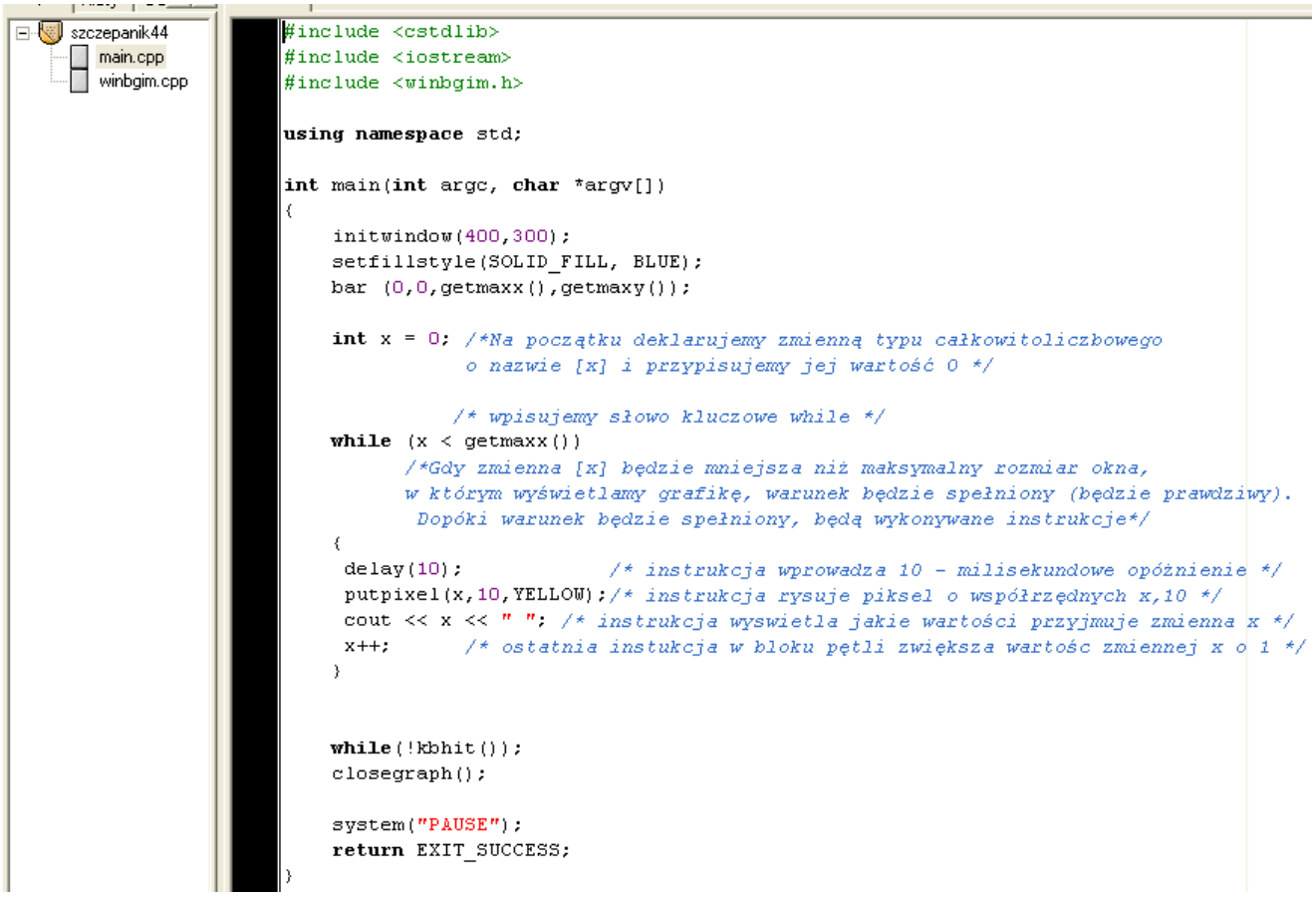
Ćwiczenie 1

1. Utwórz nowy projekt w Dev C++ i zapisz go na pulpicie w folderze o nazwie **nazwisko44**
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 – **nie przepisuj komentarza!**




```
#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());

    int x = 0; /*Na początku deklarujemy zmienną typu całkowitoliczbowego
                o nazwie [x] i przypisujemy jej wartość 0 */

              /* wpisujemy słowo kluczowe while */
    while (x < getmaxx())
          /*Gdy zmienna [x] będzie mniejsza niż maksymalny rozmiar okna,
          w którym wyświetlamy grafikę, warunek będzie spełniony (będzie prawdziwy).
           Dopóki warunek będzie spełniony, będą wykonywane instrukcje*/
    {
     delay(10);             /* instrukcja wprowadza 10 - milisekundowe opóźnienie */
     putpixel(x,10,YELLOW);/* instrukcja rysuje piksel o współrzędnych x,10 */
     cout << x << " "; /* instrukcja wyswietla jakie wartości przyjmuje zmienna x */
     x++;       /* ostatnia instukcja w bloku pętli zwiększa wartośc zmiennej x o 1 */
    }


    while(!kbhit());
    closegraph();

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

6. Z menu Uruchom wybieramy Kompiluj i uruchom
7. W oknie zapisz plik wskazujemy pulpit → folder nazwisko44 → otwieramy folder nazwisko44→ klikamy zapisz

Zwróć uwagę na wyświetlane wartości zmiennej [x].
Program wygenerował liczby od 0 do 398. Zgadza się to z warunkiem x< getmaxx().
Okno grafiki ma 400 pikseli szerokości, ale ponieważ kolejne piksele numerujemy od 0, funkcja getmaxx() zwraca wartość 399. Tak więc ostatnią wartością zmiennej [x], która będzie mniejsza od 399, jest 398.

Pętla do...while

Pętla do...while jest modyfikacją przedstawionej przed chwilą pętli while. Różnica to moment sprawdzania warunku pętli - **w pętli while warunek jest sprawdzany na początku, w pętli do...while - na końcu.** Spójrzmy na ogólny kod działania pętli typu do...while:

```
Do
{
 instrukcja 1;
 instrukcja2;
 ....
 instrukcjaN;
}
while (wyrażenie warunkowe);
```

Pętla wykonywana jest w następujący sposób:
-wykonaj instrukcje umieszczone w nawiasach klamrowych, a następnie sprawdź warunek
-jeśli warunek jest prawdziwy, ponownie wykonaj instrukcje i znowu sprawdź warunek
-gdy warunek jest fałszywy, zakończ wykonywanie pętli.

Ćwiczenie nr 2

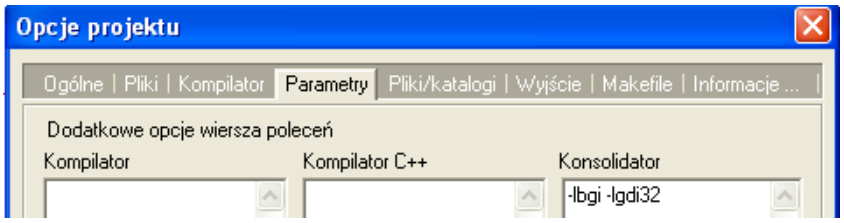
Sprawdźmy, jak będzie wyglądał i jak będzie działał poprzedni przykład, jeśli zamiast pętli while użyjemy pętli do...while.

Kod będzie bardzo podobny.

Zwróćmy szczególną uwagę na średnik, który należy umieścić na końcu linii ze słowem kluczowym while (przy pętli typu while średnika nie można postawić).

1. Utwórz nowy projekt w Dev C++ i zapisz go na pulpicie w folderze o nazwie **nazwisko45**

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 – **nie przepisuj komentarza!**

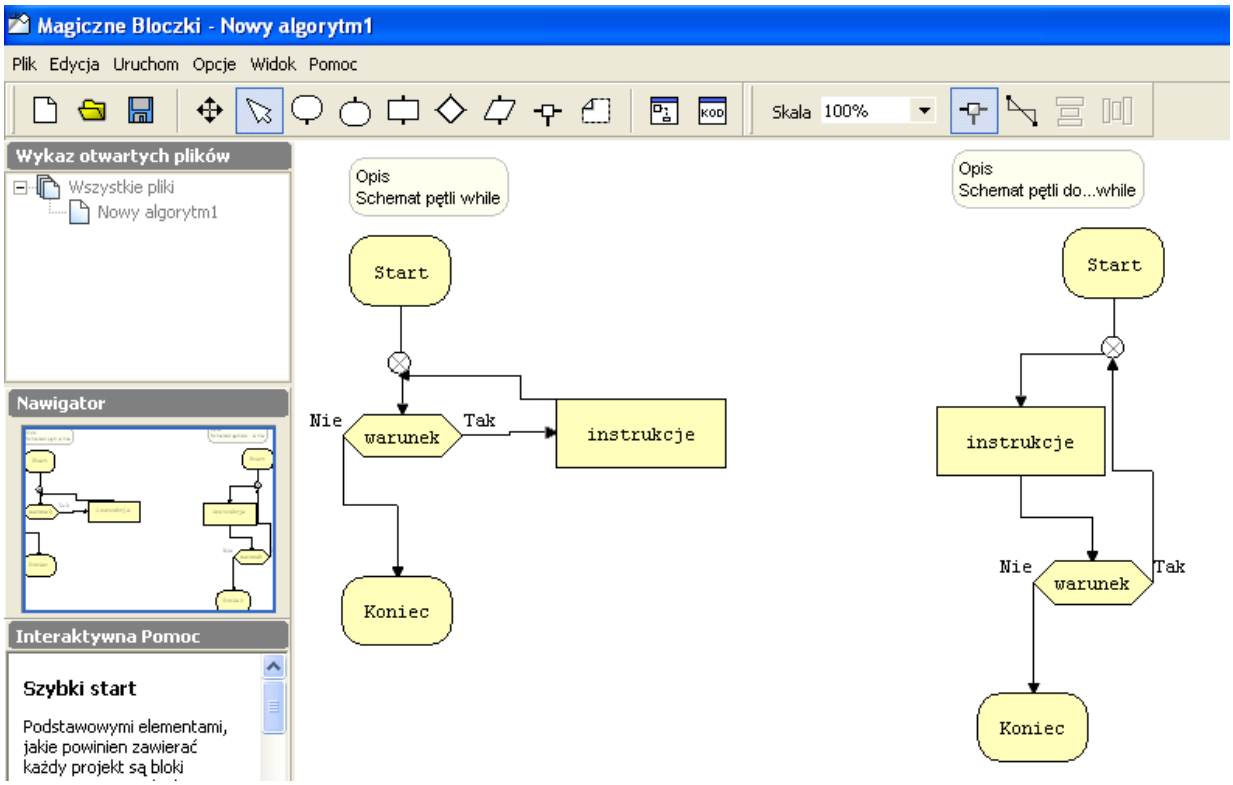
```
#include <cstdlib>
#include <iostream>
#include <winbgim.h>

using namespace std;

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

    int x = 0;
    do
    {
     cout << x << " ";
     putpixel(x,20,YELLOW);
     delay(10);
     x++;
    }
    while (x < getmaxx());


    while(!kbhit());
    closegraph();


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

6. Z menu Uruchom wybieramy Kompiluj i uruchom
7. W oknie zapisz plik wskazujemy pulpit → folder nazwisko45 → otwieramy folder nazwisko45→ klikamy zapis
8. Na ekranie zobaczymy identyczny jak w poprzednim przykładzie wynik.

Jeżeli wyrażenie warunkowe umieszczone po słowie kluczowym while będzie zawsze prawdziwe, pętla będzie wykonywana w nieskończoność. Takie działanie prowadzi najczęściej do zawieszenia programu.

Ćwiczenie3

Narysuj w magicznych bloczkach poniższe schematy w jednym oknie programu i zapisz w folderze **nazwisko45**



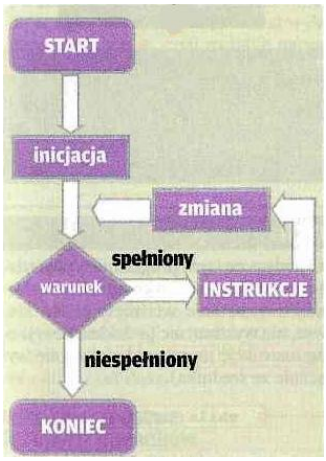
Od 1 do N, czyli pętla for

Ogólny schemat pętli typu for można przedstawić następująco:

```
for (wyrażenie inicjujące; wyrażenie warunkowe; wyrażenie zmieniające)
{
instrukcjal;
instrukcja2;
instrukcjaN;
}
```

Wykonanie instrukcji for rozpoczyna się od wyrażenia inicjującego, w którym inicjowana jest pewna zmienna (najczęściej nazywana **i** lub **j** .
Wyrażenie inicjujące wykonywane jest tylko jeden raz - na samym początku działania pętli for.
Zaraz po wykonaniu wyrażenia inicjującego oraz po udanym zakończeniu każdego przebiegu pętli sprawdzane jest wyrażenie warunkowe (zazwyczaj zawiera ono w sobie zainicjowaną przed chwilą zmienną).
Jeśli wyrażenie warunkowe zwraca wartość różną od zera (czyli jest prawdziwe), zostają wykonane instrukcje umieszczone w nawiasach klamrowych.
Jeżeli wyrażenie warunkowe zwraca wartość 0 (jest fałszywe), wykonywanie pętli zostaje przerwane.
Wyrażenie zmieniające wykonywane jest po każdym zakończeniu wykonywania instrukcji z nawiasów klamrowych, ale przed sprawdzeniem wyrażenia warunkowego.
Zadaniem wyrażenia zmieniającego jest zmiana wartości zmiennej zainicjowanej w wyrażeniu inicjującym.

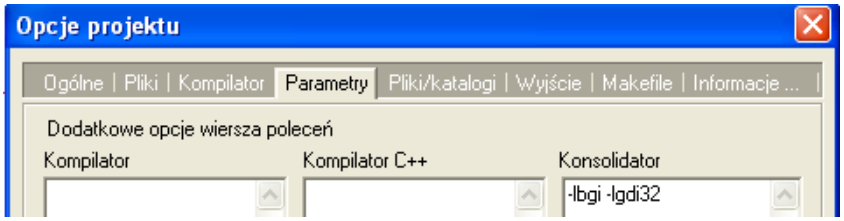
Aby lepiej zrozumieć działanie pętli for przyjrzyjmy się schematowi blokowemu



Ćwiczenie 4

1. Utwórz nowy projekt w Dev C++ i zapisz go na pulpicie w folderze o nazwie **nazwisko46**

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 – **nie przepisuj komentarza!**

```
#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 i = 0; i < getmaxx(); i++)
    /* po for wpisujemy trzy wyrazenia: inicjujące, warunkowe i zmieniające
    Zadaniem pierwszego jest inicjacja zmiennej i wartością 0.
    Drugie wyrażenie określa warunek przejścia do kolejnego przebiegu pętli -w naszym
    wypadku kolejna iteracja pętli zostaje wykonana, kiedy i jest mniejsze od maksymalnego
    rozmiaru okna (czyli gdy zapis i<xgetinaxx() jest prawdziwy).
    Wyrażenie zmieniające powoduje inkrementację (zwiększenie o 1) wartości zmiennej i .*/
    {
       delay(10);
       putpixel(i,10,YELLOW);
       cout << i << " ";
       /* Dopóki wyrażenie warunkowe jest prawdziwe, komputer wykonuje instrukcje czyli
       następuje opóźnienie (abyśmy mogli obserwować działanie pętli), zostaje postawiony
       piksel w punkcie określonym przez zmienną i liczbę 10 oraz na standardowym wyjściu
       jest wyświetlana wartość zmiennej i */
    }

    while(!kbhit());
    closegraph();


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

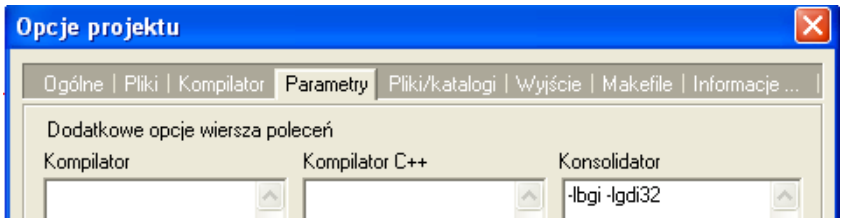
6. Z menu Uruchom wybieramy Kompiluj i uruchom
7. W oknie zapisz plik wskazujemy pulpit → folder nazwisko46 → otwieramy folder nazwisko46→ klikamy zapisz

Zagnieżdżone pętle

Pętle, podobnie jak instrukcje warunkowe, można zagnieżdżać, czyli w bloku instrukcji jednej pętli umieszczać inną pętlę. Stwórzmy więc program, który zademonstruje taką złożoną konstrukcję.

Ćwiczenie 5

1. Utwórz nowy projekt w Dev C++ i zapisz go na pulpicie w folderze o nazwie **nazwisko47**
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 – **nie przepisuj komentarza!**

```
#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 += 50)
    /*Pierwsza pętla jest wykonywana tak długo, jak długo zmienna [x] jest mniejsza od maksymalnego
    rozmiaru okna (do zmiennej x w każdym przebiegu pętli jest dodawana liczba 50 - tak więc zmienna
    x kolejno przyjmuje wartości: 0, 50, 100 i tak dalej). */
    {
        for (int r = 50; r > 0; r -= 5)
        /* W każdym przebiegu pętli 1 jest uruchamiana pętla 2. Jej zadaniem jest wyświetlanie okręgu,
        którego środek jest oddalony od lewej krawędzi okna o wartoś zmiennej x. Promień okręgu
        (zapisań w zmiennej r) w pierwszym przebiegu pętli jest równy 50, w drugim 45, potem 40, itd.*/
        {
            delay(50);
            circle(x,50,r);
        }
    }

    while(!kbhit());
    closegraph();

        system("PAUSE");
    return EXIT_SUCCESS;
}
/* Po zakończeniu wykonywania pętli 2(czyli po narysowaniu 10 okręgów o różnych promieniach) pętla 1
rozpoczyna swój drugi przebieg, w którym znowu jest rysowanych 10 okręgów. */
```

6. Z menu Uruchom wybieramy Kompiluj i uruchom
7. W oknie zapisz plik wskazujemy pulpit → folder nazwisko47 → otwieramy folder nazwisko47→ klikamy zapisz

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