Część XII C++

Warunki zagnieżdżone

Czasami zachodzi konieczność wykonania różnych instrukcji w zależności od spełnienia nie jednego, ale kilku warunków. To tak, jakbyśmy kupowali upominek - na początku sprawdzamy, jaką sumę pieniędzy możemy na niego przeznaczyć i w zależności od niej podejmujemy decyzję, jaki to ma być rodzaj prezentu: mniejszy od 50 zł, 50-100 zł czy 100 – 200 zł.
W programowaniu występowanie instrukcji warunkowych wewnątrz innych instrukcji warunkowych nazywamy zagnieżdżaniem warunków.

Ćwiczenie 1 – utworzyć program proszący użytkownika o podanie liczby podzielnej przez 2 i 3 i sprawdzeniu czy podana liczba jest rzeczywiście podzielna przez 2 i 3.

1. Utwórz nowy projekt w Dev C++ i zapisz go w folderze **nazwisko40**
2. Wprowadź do projektu modyfikacje tak aby wyglądał jak poniżej **nie przepisując komentarzy!**
3. Skompiluj i uruchom program
4. Przeanalizuj program

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

using namespace std;

int main(int argc, char *argv[])
{
    int liczba;
    /*prosimy o podanie liczby pozielnej przez 2 i 3*/
    cout << "Podaj liczbe podzielna przez 2 i 3: ";
    /*pobieramy liczbę z klawiatury*/
    cin >> liczba;
    /*sprawdzamy czy liczba jest podzielna przez 2*/
    if (liczba % 2 == 0)
    {   /* jeśli liczba jest podzielna przez 2 wykonujemy kolejną instrukcje warunkową
    której zadaniem jest sprawdzenie czy liczba jest podzielna również przez 3 */
       if (liczba % 3 == 0) cout << "Liczba jest podzielna przez 2 i 3";
       /* gdy oba warunki sa prawdziwe wyświetlamy komunikat że Liczba jest podzielna przez 2 i 3 */
       /* jesli spelniony jest pierwszy warunek podzielnośc przez 2 wyswietlamy komunikat*/
       else cout << "Liczba jest podzielna przez 2, ale nie przez 3";

    }   /* jeśli nie jest spełniony pierwszy warunek, sprawdzamy czy liczba jest podzielna przez 3 */
    else if (liczba % 3 == 0)
    {   /*jesli jest podzielna przez 3 wyswietlamy komunikat*/
       cout << "Liczba jest podzielna przez 3, ale nie przez 2";
    }
    else
    {    /* jesli nie jest podzielna przez 3 wyswietlamy komunikat */
        cout << "Liczba nie jest podzielna ani przez 2, ani przez 3";
    }


    cout << endl << endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

%

Operatory logiczne

W ćwiczeniu 1(folder nazwisko 40) do sprawdzenia dwóch warunków (podzielności liczby przez 2 i 3) wykorzystano dwie instrukcje warunkowe. Rozwiązanie takie, niestety, nie jest ani optymalne, ani czytelne. Na szczęście dzięki istnieniu operatorów logicznych można wyrażenia warunkowe łączyć ze sobą i na przykład jednocześnie sprawdzić, czy liczba podzielna jest przez 2 i przez 3.

Jeden z operatorów logicznych już wykorzystywaliśmy, był nim operator logicznej negacji oznaczany znakiem wykrzyknika !. Do tej pory jego użycie sprowadzało się do umieszczania go przed znakiem równości, co w konsekwencji tworzyło operator „różny od", czyli „nie równy".

Za pomocą operatora ! możemy również zmieniać wynik relacji. Oto przykład:
(12 == 4* 3) -prawda
!(12 ==4*3) -fałsz

Oprócz operatora negacji warto poznać jeszcze dwa:

- operator sumy logicznej oznaczany symbolem ||
- operator iloczyny logicznego oznaczany symbolem &&.

Zasada działania pokazana jest w tabeli

Operator sumy i iloczynu logicznego

wyrażenie A	wyrażenie B	wyrażenie A || B	wyrażenie A && B
prawda	prawda	prawda	prawda
prawda	fałsz	prawda	fałsz
fałsz	prawda	prawda	fałsz
fałsz	fałsz	fałsz	fałsz

Operator sumy logicznej || można utożsamiać ze słowem LUB. Dla przykładu, wyrażenie A || B jest prawdziwe wtedy i tylko wtedy, gdy A jest prawdziwe lub B jest prawdziwe lub A i B są prawdziwe. Tak więc prawdziwość dowolnego wyrażenia powoduje, że suma logiczna jest prawdziwa.

Operator iloczynu logicznego && można z kolei traktować jako spójnik I.
Wyrażenie A && B jest prawdziwe tylko wtedy, gdy prawdziwe jest zarówno A, jak i B.

Ćwiczenie 2 – cel ćwiczenia wykorzystać operator sumy i iloczynu logicznego w programie który prosi użytkownika o podanie liczby podzielnej przez 2 i 3 i sprawdza czy podana liczba jest rzeczywiście podzielna przez 2 i 3.

1. Utwórz nowy projekt w Dev C++ i zapisz go w folderze **nazwisko41**
2. Wprowadź do projektu modyfikacje tak aby wyglądał jak poniżej **nie przepisując komentarzy!**
3. Skompiluj i uruchom program
4. Przeanalizuj program

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

using namespace std;

int main(int argc, char *argv[])
{
    int liczba;
    cout << "Podaj liczbe podzielna przez 2 i 3: ";
    /*prosimy o podanie liczby podzielnej przez 2 i 3*/
    cin >> liczba;
    /*pobieramy liczbe z klawiatury*/

    if ((liczba % 2 == 0) && (liczba % 3 == 0)) cout << "Liczba jest podzielna przez 2 i 3";
    /* sprawdzamy czy liczba jest podzielna zarówno przez 2 jak i 3 , jesli tak wyświetlamy komunikat */
    else if ((liczba % 2 == 0) && (liczba % 3 != 0)) cout << "Liczba jest podzielna przez 2 ale nie przez 3";
    /* gdy nie jest spełniony pierwszy warunek, sprawdzamy, czy liczba podzielna jest przez 2 i jednoczesnie
    nie jest podzieln przez 3, jesli tak jest wyswietlamy komunikat */
    else if ((liczba % 2 != 0) && (liczba % 3 == 0)) cout << "Liczba jest podzielna przez 3 ale nie przez 2";
    /* sprawdzamy czy liczba nie jest podzielna przez 2, ale jest podzielna przez 3 */
    else cout << "Liczba nie jest podzielna ani przez 2, ani przez 3";
    /* jesli żaden z wymienionych warunków nie jest spełniony wyswietlamy komunikat*/

    cout << endl << endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Warunek znacznie uproszczony

C+ + istnieje dodatkowy, bardzo ciekawy sposób zapisu instrukcji warunkowej if...else:
warunek? wyrażenie 1: wyrażenie 2
Taki zapis można rozszyfrować następująco:
Jeśli spełniony jest warunek, wykonywane jest wyrażenie 1, gdy warunek jest fałszywy, zostaje wykonane wyrażenie 2.

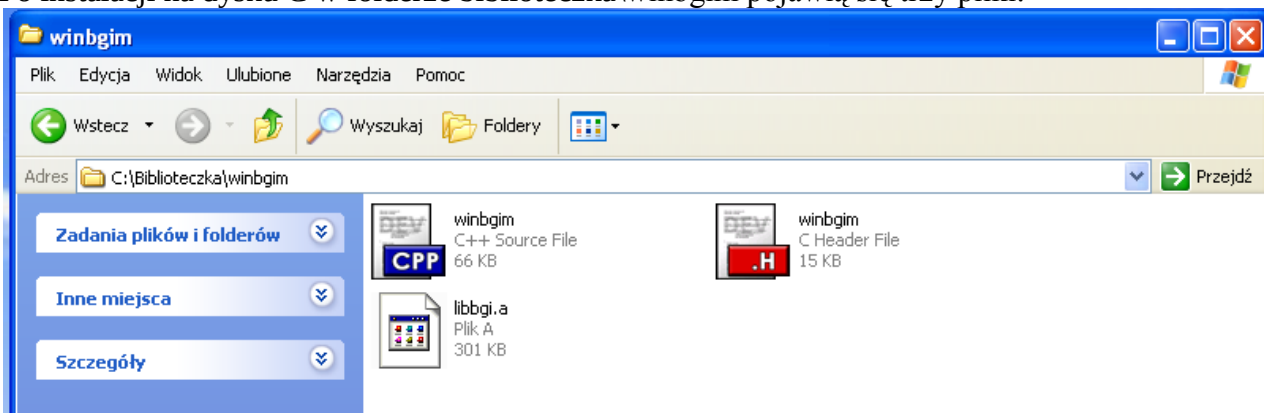
Ćwiczenie 3

1. Utwórz nowy projekt w Dev C++ i zapisz go w folderze **nazwisko42**
2. Wprowadź do projektu modyfikacje tak aby wyglądał jak poniżej **nie przepisując komentarzy!**
3. Skompiluj i uruchom program
4. Przeanalizuj program
5. Sprawdź działanie programu dla różnych liczb (tylko sprawdź)

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

using namespace std;

int main(int argc, char *argv[])
{
    int liczba;
    char dwa, trzy;
    cout << "Podaj liczbe: ";
    cin >> liczba;
    /* po pobraniu wartości wykonujemy  pierwszą instrukcję. Jej działanie jest  takie:
     Jeżeli reszta z dzielenia  wartości przez liczbę 2 jest równa 0,  wykonywane
     jest przypisanie  do zmiennej [dwa] znaku T,  w przeciwnym wypadku do zmiennej
     |dwa| przypisywany  jest znak N. */
    (liczba % 2 == 0) ? dwa = 'T' : dwa = 'N';
    /* instrukcja druga jest trochę inną formą krótkiego zapisu instrukcji warunkowej.
    W tym wypadku do zmiennej [trzy]  przypisujemy znak T wtedy, gdy reszta z dzielenia liczby
    przez 3 jest równa 0, lub znak N, gdy reszta jest różna od 0.  */
    trzy = (liczba % 3 == 0) ? 'T' : 'N';

    cout << "Czy liczba jest podzielna przez dwa? " << dwa << endl;
    cout << "Czy liczba jest podzielna przez trzy? " << trzy << endl;


    cout << endl << endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Grafika w C++

Aby można korzystać z prostych instrukcji umożliwiających rysowanie na ekranie punktów, linii czy okręgów, należy zainstalować biblioteki WinBGIm.
Po instalacji na dysku C w folderze biblioteczka\winbgim pojawią się trzy pliki.



Kopiujemy je do odpowiednich katalogów.

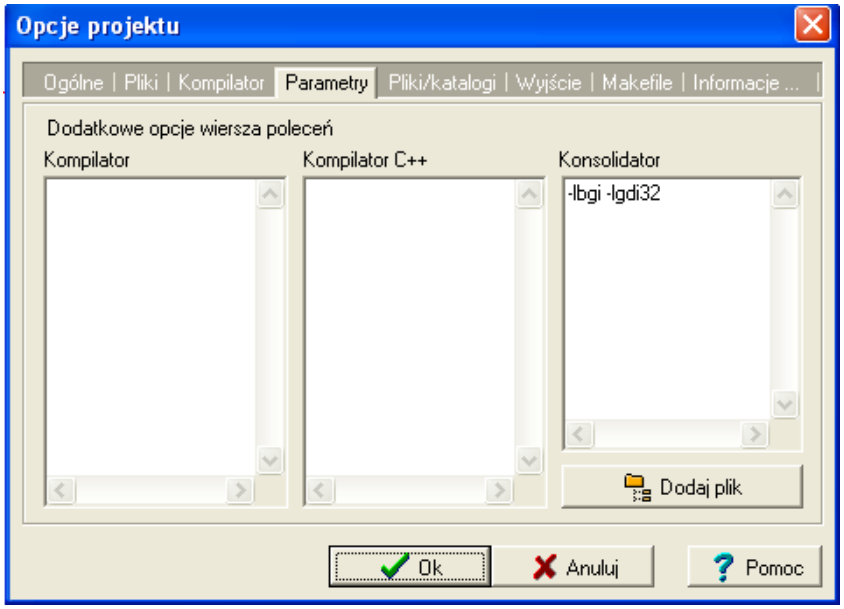
Pliki winbgim.cpp (niebieski) i winbgim.h (czerwony) kopiujemy do katalogu c:\dev-cpp\include.

Plik lilibbgi.a kopiujemy do katalogu c:\dev-cpp\lib.

Nie kopiuj gdyż wszystkie powyższe operacje zostały już przeprowadzone przez administratora !!!

Ćwiczenie 4

1. Utwórz nowy projekt w Dev C++ i zapisz go na pulpicie w folderze o nazwie **nazwisko43**
2. Z menu Projekt wybieramy Opcje projektu i w oknie klikamy na zakładkę Parametry W pole konsolidator wpisujemy:
   myślnik, wyraz lbgi, odstęp(spacja), myślnik, wyraz 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> /* to polecenie umozliwia rysowanie punktów lub linii */

using namespace std;

int main(int argc, char *argv[])
{
    initwindow(400,300); /* ta instrukcja wyswietla okno o rozdzielczości 400 na 300 */
    setfillstyle(SOLID_FILL, BLUE); /* wypełnienie całego okna kolorem niebieskim */
    bar (0,0,getmaxx(),getmaxy());

    line(50,50,350,250); /* instrukcja rysująca linię od punktu 50,50 do punktu 50,250 */

    while(!kbhit()); /* ta instrukcja i ponizsza wstrzymuja zamknięcie okna z grafiką */
    closegraph();

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

6. Z menu Uruchom wybieramy Kompiluj i uruchom

7. W oknie zapisz plik wskazujemy pulpit → foldernazwisko43 → otwieramy folder nazwisko43→ klikamy zapisz

Niektóre przydatne polecenia z biblioteki WinBGIm

polecenie	opis
delay (x)	opóźnienie wykonywania programu o x milisekund
initwindow (x, y)	uruchamia graficzne okno o rozmiarach x pikseli na y pikseli
setfillstyle (styl, kolor)	ustawia styl i kolor wypełnienia
bar (x1, y1, x2, y2)	rysuje prostokąt rozciągnięty od punktu x1,y1 do punktu x2,y2
putpixel (x, y, kolor)	rysuje punkt o współrzędnych x,y i ustawionym kolorze
getmaxx ()	pobiera rozmiar okna w poziomie
getmaxy ()	pobiera rozmiar okna w pionie
circle (x, y, r)	w punkcie x,y rysuje okrąg o promieniu r

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