Część IX C++

Jak napisać program obliczający pola powierzchni różnych figur płaskich?

Na początku, przed stworzeniem właściwego kodu programu zaprojektujemy naszą aplikację i stworzymy schemat blokowy prezentujący jej działanie.

Określamy cel
Chcemy napisać program, więc musimy wiedzieć, co będzie robił.
Określamy w punktach jego założenia:

- program będzie obliczał pola powierzchni figur: prostokąta, trójkąta i koła,
- wszystkie niezbędne do obliczenia dane będzie można wprowadzić z klawiatury,
- kolejne pola będą obliczane po kolei,
- na końcu porównamy obliczone pola figur i wyświetlimy różne relacje.

Piszemy algorytm
Wiemy, co będzie robił program. Postawmy więc sobie pytanie, jak będzie to robił. Napiszmy jego algorytm - najlepiej w punktach.

W opisowym języku polecenia algorytm dla komputera mogłyby wyglądać tak:

- pobierz dane o długości boków prostokąta i zapisz je w zmiennych [a] i [b],
- oblicz pole powierzchni prostokąta i zapisz w zmiennej [pole prostokąta] i wyświetl wynik na ekranie,
- pobierz długości podstawy i wysokości trójkąta i zapisz je w zmiennych [x] i [h],
- oblicz pole powierzchni trójkąta, zapisz w zmiennej [pole trójkąta] i wyświetl wynik na ekranie,
- pobierz długość promienia koła i zapisz w zmiennej [r],
- oblicz pole koła, zapisz w zmiennej [pole koła] i wyświetl wynik,
- porównaj obliczone pola i wyświetl - wyniki relacji w postaci zer i jedynek

Schemat blokowy
Ćwiczenie 1 – stwórz schemat w magicznych bloczkach i zapisz w folderze nazwisko29

Ćwiczenie 2

1. Utwórz nowy projekt w Dev C++ i zapisz go w folderze nazwisko29
2. Wprowadź do projektu modyfikacje tak aby wyglądał jak poniżej
3. Skompiluj i uruchom program
4. Przeanalizuj program

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

using namespace std;

int main(int argc, char *argv[])
{
    const int PI = 3;
    int a, b, x, h, r, pole_prostokata, pole_trojkata, pole_kola;

    cout << "Podaj dlugosc bokow prostokata: ";
    cin >> a >> b;
    pole_prostokata = a * b;
    cout << "Pole prostokata o bokach dlugosci " << a << " i " << b;
    cout << " wynosi " << pole_prostokata << endl << endl;

    cout << "Podaj dlugosc podstawy oraz wysokosc trojkata: ";
    cin >> x >> h;
    pole_trojkata = x * h / 2;
    cout << "Pole trojkata, w ktorym podstawa ma " << x << " a wysokosc " ;
    cout << h << " wynosi " << pole_trojkata << endl << endl;

    cout << "Podaj promien kola: ";
    cin >> r;
    pole_kola = PI * r * r;
    cout << "Pole kola o promieniu " << r << " wynosi " << pole_kola;
    cout << endl << endl;

    cout << "Pole prostokata jest wieksze od pola kola? Odpowiedz: ";
    cout << (pole_prostokata > pole_kola) << endl;

    cout << "Pole kola jest rozne od pola trojkata? Odpowiedz: ";
    cout << (pole_kola != pole_trojkata) << endl;

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

Przeanalizuj program!!!

Typy danych

Każda zmienna może się zmieniać tylko w określonym zakresie.
Do tej pory operowaliśmy **zmiennymi całkowitoliczbowymi** (zapewniało nam to słowo kluczowe [int] przed nazwą zmiennej w jej deklaracji).
Zmienne całkowitoliczbowe mają spore ograniczenia i dokładne obliczenie (na przykład pola koła) z ich wykorzystaniem nie są możliwe.

Występujące w C++ typy danych można podzielić na trzy rodzaje:

- arytmetyczne,
- logiczne,
- inne.

Do typów arytmetycznych należą typy całkowitoliczbowe, zmiennopozycyjne oraz znakowe. Wśród innych typów możemy wyróżnić typ nieokreślony, wskaźnikowy, referencyjny, wyliczeniowy, tablicowy.

Wszystkie wymienione typy danych są wbudowane w język C+ + . Oznacza to, że określone są przez twórców tego języka. Oprócz wbudowanych typów danych można tworzyć swoje własne typy danych. Przykładem może być typ string (poznaliśmy go przy okazji omawiania stałych). Zdefiniowany on jest w jednej z bibliotek i mogliśmy z niego korzystać tylko dzięki temu, że dołączyliśmy tę bibliotekę do naszego kodu za pomocą dwóch dyrektyw #include, które są automatycznie umieszczane przez aplikację Dev-C++ w szkielecie programu

Liczby całkowite

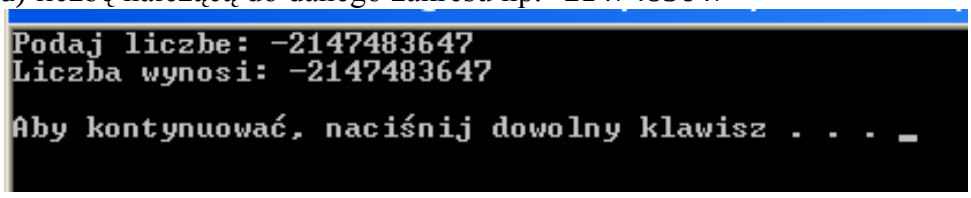
Wpisując w deklaracji zmiennej przed jej nazwą słowo kluczowe [int], ustalamy, że zmienna ta będzie typu całkowitoliczbowego, czyli będzie mogła przechowywać liczby całkowite. Jest to jednak duże uproszczenie.
Tak naprawdę zmienna zadeklarowana jako **|int|** może przechowywać liczby całkowite ze znakiem (czyli liczby ujemne i dodatnie) od **-2147483648 do 2147483647**.

Ćwiczenie 3

1.Utwórz nowy projekt o nazwie **nazwisko30**
2.Wpisz poniższy kod programu, który pobiera z klawiatury wartość i zapisuje ją w zmiennej [liczba]

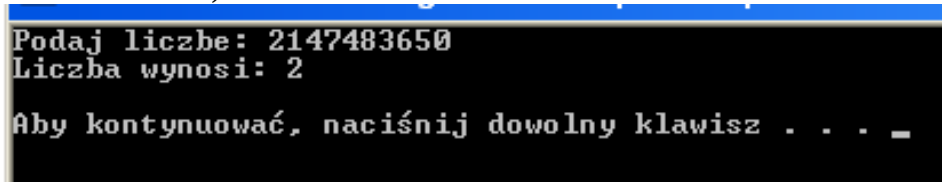
```
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
    int liczba;
    cout << "Podaj liczbe: ";
    cin >> liczba;

    cout << "Liczba wynosi: " << liczba << endl;

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

3.Po skompilowaniu i uruchomieniu programu wpiszmy:
a) liczbę należącą do danego zakresu np. -2147483647



Liczba została poprawnie przypisana do zmiennej

b) liczbę nie należącą do danego zakresu np. 2147483650 (uruchom jeszcze raz plik nazwisko30.exe)



Liczba nie została przypisana do zmiennej !!!

Dzięki tak zwanym **modyfikatorom** możemy nieco wpływać na zakres danych, które można zapisać w zmiennej [int] (oraz w zmiennych innych typów).

Modyfikatory to nic innego jak określenia różnych odmian jednego typu. Oto najważniejsze z modyfikatorów:

- **short** - ang. krótki - pozwala na zmniejszenie zakresu liczb, a dzięki temu na zmniejszenie ilości pamięci wykorzystywanej do przechowywania zmiennej,
- **long** - ang. długi - dzięki niemu zwiększymy zakres liczb, które mogą być przechowywane w zmiennej,
- **signed** - określa, że dany typ reprezentuje dane dodatnie i ujemne (ze znakiem -),
- **unsigned** - określa, że liczby przechowywane w zmiennej tego typu mogą być tylko dodatnie.

Jak korzystać z modyfikatora? Wystarczy napisać go przed słowem kluczowym |int|. W tabelce przedstawiono wszystkie możliwe kombinacje modyfikatorów typu [int]. W tabeli znajdują się również najczęściej występujące zakresy liczb, które można przechowywać w zmiennej określonego typu całkowitoliczbowego.

Typy całkowitoliczbowe

definicja	zakres	
	od	do
unsigned short int	0	65535 (2^{16}-1)
signed short int	-32768 (-2^{15})	32767 (2^{15}-1)
short int	-32767 (-2^{15}+1)	32768 (2^{15})
unsigned int	0	4294967295 (2^{32}-1)
signed int	-2147483647 (-2^{31}+1)	2147483648 (2^{31})
int	-2147483648 (-2^{31})	2147483647 (2^{31}-1)
unsigned long int	0	4294967295 (2^{32}-1)
signed long int	-2147483647 (-2^{31}+1)	2147483648 (2^{31})
long int	-2147483648 (-2^{31})	2147483647 (2^{31}-1)

Liczby z przecinkiem

Liczby całkowite nie pozwalają na przeprowadzanie dokładnych obliczeń (na przykład wspomnianego już wcześniej pola powierzchni koła).

Do tego celu służą **liczby zmiennopozycyjne** (zwane również **zmiennoprzecinkowymi)**, czyli takie, które pozwalają na zapisanie części ułamkowej.

W języku C+ + istnieją trzy typy, które pozwalają na zapisanie takich właśnie liczb:

- **float** - liczby zmiennopozycyjne z pojedynczą precyzją,
- **double** - ang. podwójne - liczby zmiennopozycyjne z podwójną precyzją,
- **long double** - liczby zmiennopozycyjne z rozszerzoną precyzją.

Dopisanie modyfikatora long nie wpływa na rozszerzenie zakresu liczb. Jednak na starszych pecetach i kompilatorach, typ krótki i zwykły mają mniejszy zakres (na przykład ten pierwszy od -127 do 128, a drugi od -32767 do 32768), więc typ długi pozwala rzeczywiście na zapisanie większej liczby

Typy zmiennopozycyjne

definicja	zakres	
	od	do
float	$3{,}4 * 10^{-38}$	$3{,}4 * 10^{38}$
double	$1{,}7 * 10^{-308}$	$1{,}7 * 10^{308}$
long double	$3{,}4 * 10^{-4932}$	$3{,}4 * 10^{4932}$

Ttyp zmiennoprzecinkowy pozwala na dokładne obliczenia.

Ćwiczenie 4

Cel ćwiczenia: napisać kod będący rozszerzeniem fragmentu poprzednio stworzonego programu. **Jego zadaniem będzie dokładne obliczenie pola koła** (wynik działania poprzedniego kodu był zawsze liczbą całkowitą).

1. Utwórz nowy projekt w Dev C++ i zapisz go w folderze **nazwisko31**
2. Wprowadź do projektu modyfikacje tak aby wyglądał jak poniżej
3. Skompiluj i uruchom program
4. Przeanalizuj program

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

using namespace std;

int main(int argc, char *argv[])
{
    const float PI = 3.141592;
    float r, pole_kola;

    cout << "Podaj promien kola: ";
    cin >> r;
    pole_kola = PI * r * r;
    cout << "Pole kola o promieniu " << r << " wynosi " << pole_kola;

    cout << endl << endl;

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

Liczby zmiennoprzecinkowe można zapisywać na dwa sposoby:

- pierwszy sposób, znany wszystkim z matematyki, polega na zapisaniu części całkowitej **liczby i po kropce** jej części ułamkowej (np. 34.768). **Część całkowitą od ułamkowej oddzielamy kropką, a nie przecinkiem**
- drugi sposób zapisu, wygodny w przypadku bardzo małych lub bardzo dużych liczb to notacja wykładnicza (np. 2.7e4)

Ćwiczenie 5

1. Utwórz nowy projekt w Dev C++ i zapisz go w folderze **nazwisko32**
2. Wprowadź do projektu modyfikacje tak aby wyglądał jak poniżej
3. Skompiluj i uruchom program
4. Przeanalizuj program

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

using namespace std;

int main(int argc, char *argv[])
{
    float liczba1 = 34.768;
    float liczba2 = 2.7e4;

    cout << liczba1 << endl << liczba2 << endl << endl;


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

```
34.768
27000

Aby kontynuować, naciśnij dowolny klawisz . . .
```

2.7e4 = 2,7 * 10^4 = 2,7 * 10000=27000

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