Zauważyłam sprawdzając wasze prace, że mieliście problem ze schematem blokowym, dlatego pewne rzeczy wytłumaczymy.

**Schemat blokowy –** przedstawia algorytm rozwiązania problemu w postaci symboli graficznych i przedstawia szczegółowo wszystkie operacje arytmetyczne, logiczne, sterujące wraz z kolejnością ich wykonywania. Schemat blokowy składa się z wielu elementów, kilka z nich przedstawiam poniżej w tabelce:

|  |  |
| --- | --- |
| Wygląd elementu | Opis |
|  | **Bloki graniczne** – **start** to początek algorytmu, ma tylko jedno połączenie wychodzące, **stop** to koniec algorytmu, ma tylko jedno połączenie wchodzące. **Start** może być tylko jeden raz w schemacie, **stop** może być więcej jak jeden raz. |
|  | **Połączenie** pomiędzy blokami, określa kolejność wykonywanych instrukcji |
|  | **Blok kolekcyjny –** łączy kilka różnych dróg algorytmu |
| X := 10 | **Blok operacyjny –** operacja lub blok operacji, w których wyniku ulega zmianie wartość zmiennej lub zmiennych. Blok operacyjny ma kształt prostokąta, na schematach blokowych operację przypisania piszemy „:=” (dwukropek i znak równości), w programach C++ operacja przypisania to tylko znak równości (=). Jak to czytamy? Zmiennej x przypisz 10 (lub dziesięć umieść w zmiennej x). Wchodzi jedno połączenie, wychodzi też tylko jedno |
| Wypisz: x | **Blok wejścia/wyjścia –** czyli operacje wprowadzania i wyprowadzania danych ( metody: cout<< i cin>>). Wchodzi jedno połączenie i wychodzi jedno. Blok ma kształt równoległoboku |
| X == 5  Tak  Nie | **Blok decyzyjny –** określa wybór jednej z dwóch możliwych dróg przebiegu algorytmu. Zawsze pójdziemy tylko jedną drogą. W bloku pytamy „czy x==5” czy x równa się 5? Jeśli tak, to pójdziemy w prawo i w dół, jeśli nie to w lewo. Nigdy przy Bolku decyzyjnym nie wykonają się wszystkie instrukcje, bo przecież x nie może być jednocześnie równy 5 i nie równy 5. Tu jest albo w lewo, albo w prawo. Dlatego nazywa się blokiem decyzyjnym. Jak widać 1 połączenie wchodzi do bloku decyzyjnego, a 2 wychodzą. |
|  | Opisz ten blok |
|  | Opisz ten blok |

Przykład:

Zapiszemy specyfikację problemu, schemat blokowy i kod programu, który oblicza średnią arytmetyczną dwóch liczb podanych przez użytkownika i wyświetla wynik na ekranie.

Specyfikacja :

**Problem algorytmiczny:**  Obliczanie średniej arytmetycznej dwóch liczb rzeczywistych i

wypisanie wyniku na ekranie

**Dane wejściowe: a, b ϵ R**

**Dane wyjściowe: sr ϵ R – średnia liczb a i b**

Teraz zapiszemy algorytm obliczania średniej w postaci schematu blokowego:

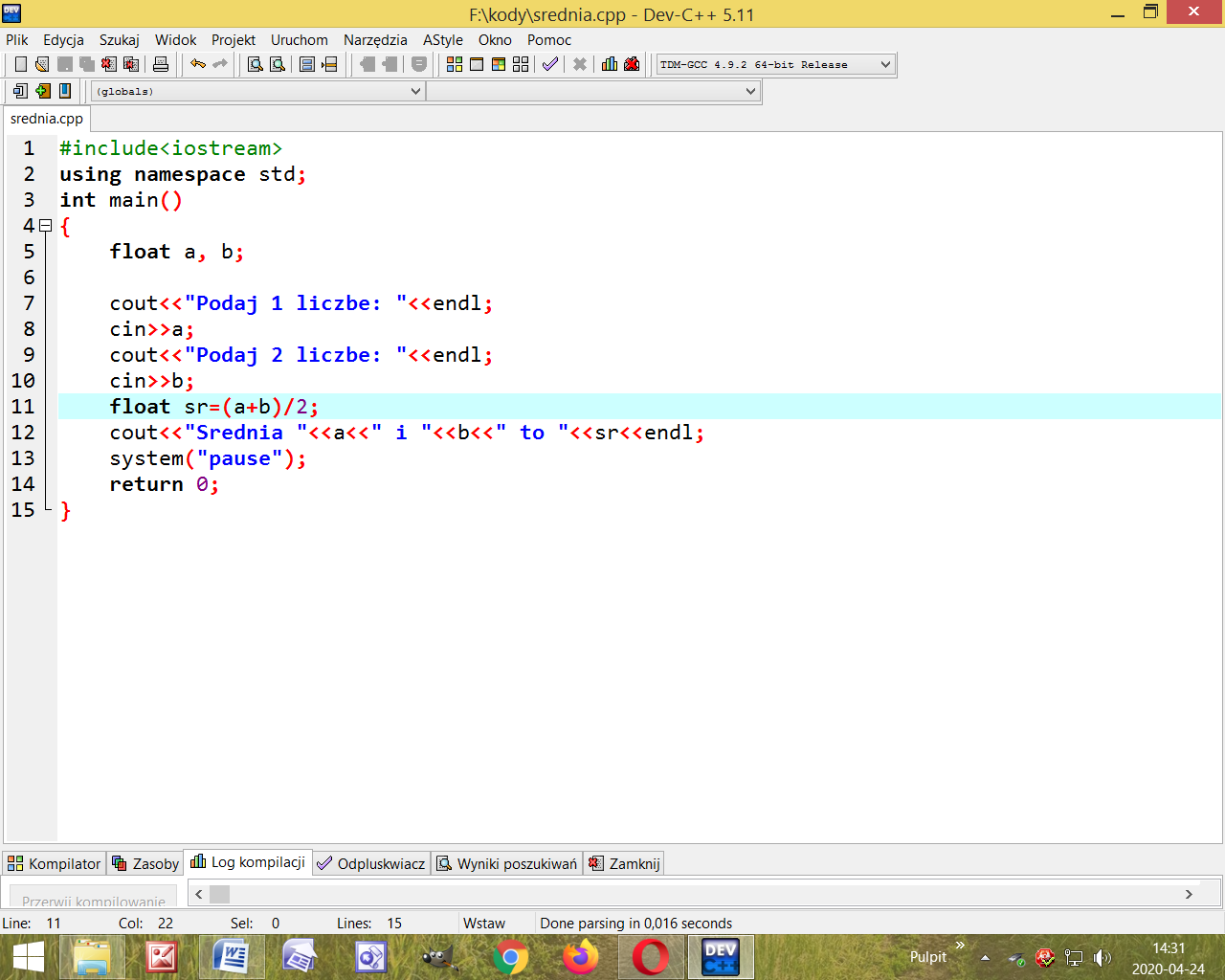
Wczytaj: a, b

sr := (a+b)/2

Wypisz: sr

Ten sam algorytm zapiszemy w języku C++. Kolejność działań:

1. Włączamy Dev-C++, wybieramy z menu górnego: Plik->Nowy->Plik źródłowy
2. Dołączamy bibliotekę wej/wyj, czyli #include<iostream>
3. Dołączamy przestrzeń nazw, aby nie pisać std::cout<<”Tekst”, tylko cout<<”Tekst”
4. Piszemy int main(), czyli nazwę funkcji głównej programu i otwieramy ją znakiem {
5. Deklarujemy 2 zmienne, do których użytkownik wprowadzi jakieś liczby float a, b;
6. Możemy zadeklarować dodatkowo zmienną srednia i przypisać jej wyrażenie sr=(a+b)/2 lub obliczyć to „w locie” poprzez zapis cout<<”Srednia = „<<(a+b)/2<<endl;
7. Dołączamy system(„pause”); aby obraz się zatrzymał oraz return 0; bo funkcja główna programu zwraca po wykonaniu się liczbę całkowitą typu int (Piszemy int main()), zwróci zero jak się poprawnie wykona, zwróci -1 jak będzie błąd i wtedy otrzymamy od systemu jakiś komunikat o błędzie.
8. Zamykamy funkcję główną znakiem } i możemy kompilować.
9. W Dev-C++ naciskamy klawisz funkcyjny F11 (kompiluj i uruchom). Przed kompilacją program poprosi nas o zapisanie pliku. Nadajemy jakąś nazwę, np.: cw1, w dolnym okienku Pliki typu: wybieramy C++ source files i plik zapisze nam się z rozszerzeniem cpp, co oznacza że został napisany w języku C++. Jeśli wszystko zostało dobrze napisane program zostanie skompilowany (czyli przetłumaczony na język zrozumiały dla procesora) i wykonany w konsoli. Jeśli będą błędy linia kodu z błędem lub pod linią z błędem podświetli się na czerwono. Należy znaleźć błąd i znowu F11, aż do poprawnego wykonania. Po poprawnym skompilowaniu w folderze gdzie zapisaliście plik pojawi się drugi plik z rozszerzeniem exe. Możemy ten plik wykonywać tak jak każdy inny program. /czyli podwójne kliknięcie uruchamia.
10. Tak wygląda poprawny kod:



Niektórzy pisali w pracach end1(jedynka za literą d), zamiast endl, co oznacza koniec linii *end line* tak jak enter w Word, przenosi do nowej linii;

Jeśli zmienne są tego samego typu, możemy napisać 1 raz typ, a potem oddzielic ich nazwy przecinkiem. Jeśli maja być różnych typów, musimy zrobić tak:

float a, b, c, srednia, x;

int zmienna1;

char znak;

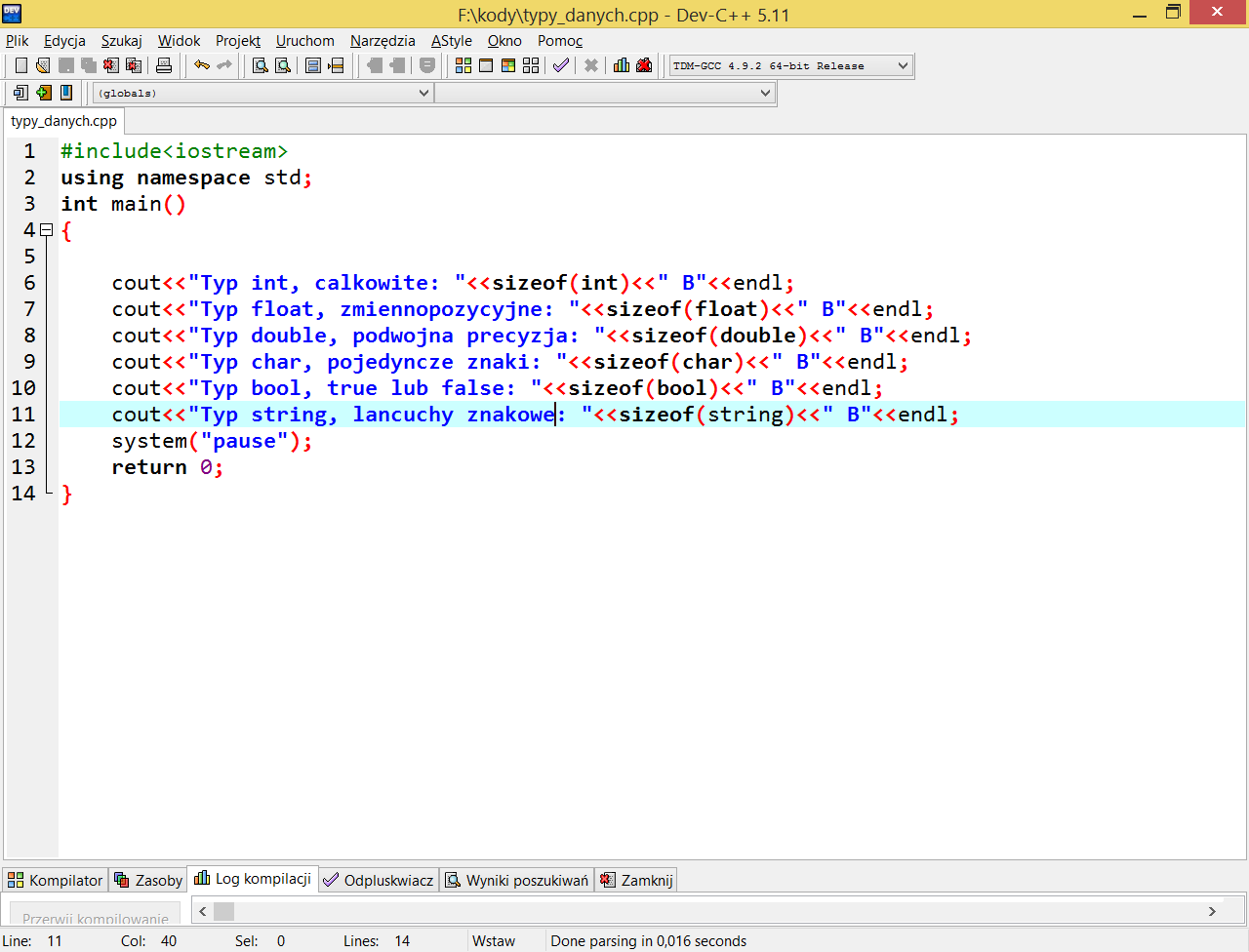
Możemy też już przy deklaracji zmiennej, jeśli jest taka potrzeba przypisać jej wartość w taki sposób:

float Pi = 3.14, suma=0;

Tekst który ma być wyświetlony na ekranie musi być ujęty w znaki cudzysłowu „”, natomiast jeśli chcemy wyświetlić wartość jaką zmienna przechowuje, to piszemy tylko jej nazwę

Dlaczego napisałam (a+b)/2, bo działania w C++ wykonują się od prawej do lewej, bez nawiasu było by tak: b podziel przez 2 i dodaj do a. Wynik fałszywy! Musi być nawias.

**Deklaracja zmiennej – co to znaczy?** Zadeklarować zmienną to znaczy zarezerwować w pamięci RAM obszar pamięci o jakiejś nazwie i rozmiarze jaki zajmuje dany typ. A ile zajmuje? To zależy od środowiska programistycznego w którym piszemy. Sprawdźmy ile Bajtów zajmują podstawowe typy danych których będziemy używać w Dev-C++. Skorzystamy z funkcji sizeof() której jako argumenty wpiszemy poszczególne typy.



Napiszcie ten kod i F11(zapisać i skompilować), zapisać w folderze.

Żeby pisać proste programy, które np. coś obliczają, musimy poznać operatory stosowane w C++. **Poniższe tabelki przepisać do zeszytu lub wydrukować, wyciąć i wkleić do zeszytu. Sprawdzę!**

Jeśli zmienne lub stałe poprawnie połączymy operatorami zgodnie z regułami matematycznymi, to otrzymamy wyrażenie. Zmienne i stałe występujące w wyrażeniu nazywamy **operandami.**

**Operatory matematyczne obowiązujące w C++**

|  |  |
| --- | --- |
| C/C++ | Znaczenie |
| + | Dodawanie |
| - | Odejmowanie |
| \* | Mnożenie |
| / | Dzielenie |
| / | Dzielenie całkowite, czyli zaokrąglenie wyniku do liczby całkowitej. Jeśli dzielimy dwie liczby zadeklarowane jako int to w wyniku dzielenia otrzymamy int. 7/2=3. Ale jeśli liczby były by typu float (z przecinkiem), to w wyniku dzielenia 7/2 będzie wynik 3.5 |
| % | Reszta z dzielenia liczb całkowitych. Reszta z dzielenia 7/2 to 1 bo w 7 mieści się 3 dwójki i zostaje 1 |

**Operatory relacji w C++**

|  |  |
| --- | --- |
| C/C++ | Znaczenie |
| == | Czy równy. Pytamy czy x jest równe 12? Napiszemy x == 12 |
| > | większy |
| >= | Większy lub równy |
| < | mniejszy |
| <= | Mniejszy lub równy |
| != | Negacja, różny, czyli nieprawda że równy. Jak napiszemy, że x ma być różne od zera? **x!=0** |

**Operator przypisania to jeden znak równości, czyli =. Jak napiszemy zmiennej xyz przypisz wartość 45? xyz = 45**

**Operatory logiczne**

|  |  |  |
| --- | --- | --- |
| C/C++ | Znaczenie | Zapis matematyczny |
| && | Koniunkcja – iloczyn zdań | ˄ |
| || | Alternatywa – suma zdań | ˅ |
| ! | Negacja - zaprzeczenia | ˜ |

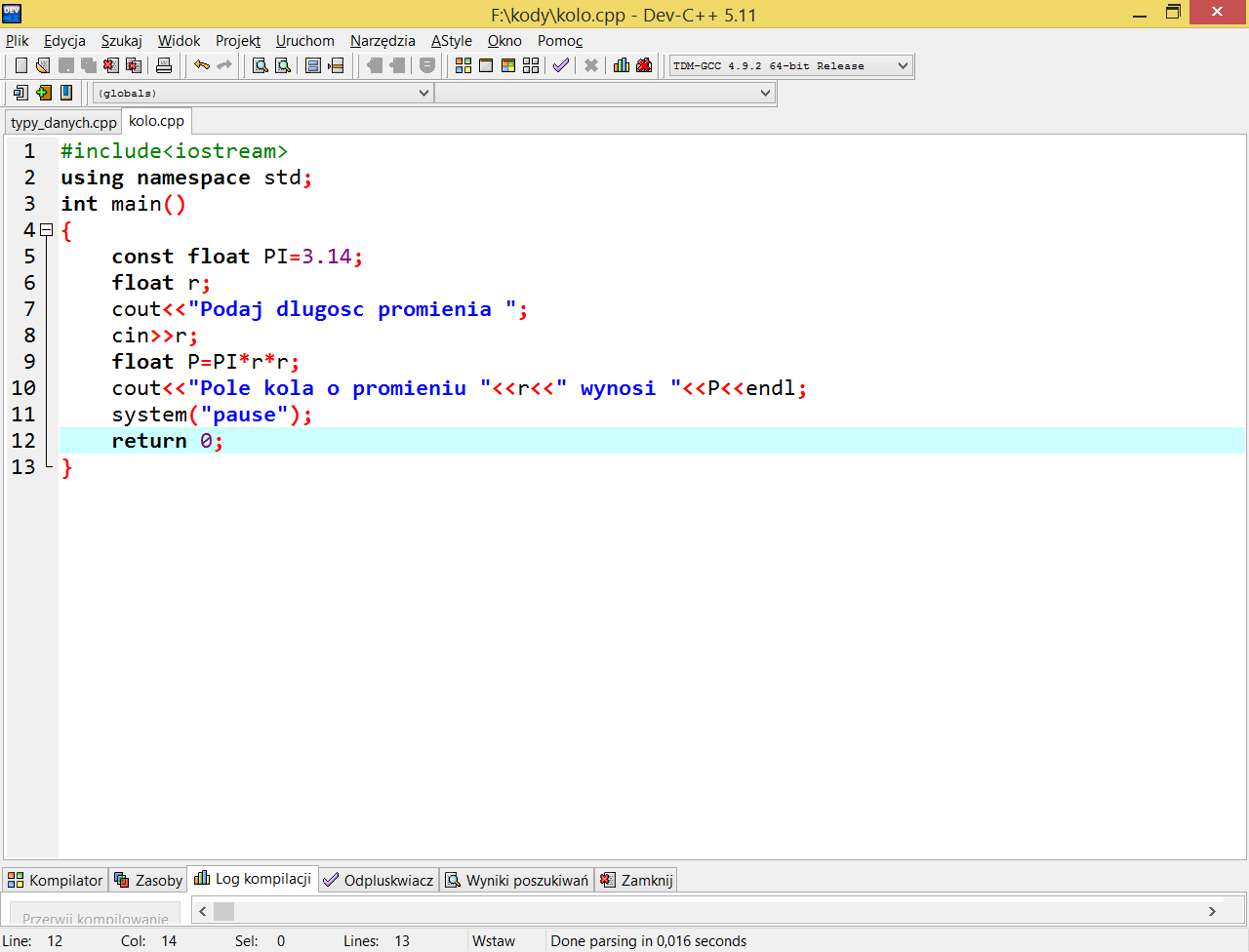
**Pamiętajmy przy deklarowaniu zmiennych o tym, że:**

* W języku C++ małe i wielkie litery w nazwach zmiennych, instrukcji, funkcji i procedur mają znaczenie. Zmienne o nazwach Zm1, zm1 to dwie różne zmienne;
* W nazwach zmiennych możemy stosować tylko małe i duże litery alfabetu angielskiego, cyfry od 0 do 9 (ale nie na pierwszym miejscu) oraz znak podkreślenia. Nie można jako nazw zmiennych stosować słów kluczowych języka oraz znaków narodowych(ąęóńćźżł).  
  **Dobre nazwy zmiennych: zm1, zm\_1, z1m,   
  Złe nazwy: zm-1, kąt, pięć, return, pierwsza zmienna – niedozwolony myślnik, litera ą, słowo kluczowe, spacja w ostatniej**
* Zmienne w C++ możemy deklarować wszędzie, ale zawsze przed pierwszym użyciem.

**Przykład 1**

Napiszmy program, który będzie nam obliczał pole koła o promieniu r. Długość promienia podaje uzytkownik z klawiatury. Program wyświetli nam pole.

Najpierw zastanawiamy się ile zmiennych potrzebujemy, żeby obliczyć pole koła, które wynosi P=PI\*r2. Wartość PI wpiszemy w programie jako stałą o wartości 3.14. Stałe deklarujemy za pomocą operatora const. Musimy zarezerwować obszar pamięci o nazwie r na przechowanie długości promienia oraz obszar pamięci o nazwie P na obliczone pole. Czyli mamy dwie zmienne zrobimy je zmiennopozycyjne, bo ktoś może podać długość 4.5 i jedną stałą PI. Pamiętajmy, że w C++ nie piszemy 3,14( z przecinkiem), tylko 3.14 (zamiast przecinka kropka). Rozwiązanie:



Dlaczego napisałam p=PI\*r\*r zamiast PI\*r2 ? Bo do potęgowania w C++ służy pewna funkcja, której jeszcze nie znacie. Nazywa się pow(co potęguję, do której potęgi), aby jej uzywać potrzeba dodać bibliotekę matematyczną na początku kodu.

Do wykonania w domu:

1. Wypełnić do końca tabelkę z blokami
2. Przepisać do zeszytu lub wydrukować i wkleić tabelki z operatorami
3. Oraz wykonać zadanie które jest poniżej

Zadanie praktyczne do wykonania w domu.

Napisz program, który będzie obliczał pole i objętość sześcianu. Bok sześcianu podaje uzytkownik z klawiatury. Po obliczeniu wyświetl pole powierzchni i objętość sześcianu.

**Do 8 maja oczekuję 3 zdjęć z wykonanej pracy**. Pierwsze – wypełniona tabelka, drugie zdjęcie notatek lub wklejonych do zeszytu tabelek z operatorami i trzecie zdjęcie kodu zrobione tak, abym widziała wasz pulpit na monitorze. Powodzenia