1 Program komputerowy-podstawowe pojęcia. Struktura programu w języku C++.

W jaki sposób zmusić maszynę, by wykonała nasze polecenia? Większość znanych nam maszyn jest po prostu tak zbudowana, by wykonywać określone czynności. Tak samo było z pierwszymi komputerami - były budowane tak, by spełniać określone zadania. Uniwersalne komputery, oparte na pomyśle Johna von Neumanna, mogły wykonywać dowolne zadania bez konieczności zmiany ich budowy. Wystarczyło zapisać w ich pamięci odpowiednie instrukcje, czyli program działania. Instrukcje muszą być wyrażone przy pomocy symboli zrozumiałych dla danej maszyny, a więc w odpowiednim języku programowania.

Program komputerowy to lista poleceń zapisana w jednym języku programowania, zgodnie z obowiązującymi w tym języku zasadami. Celem programu jest przetwarzanie danych w określonym przez twórcę zakresie. Programy (zwane Oprogramowaniem) tworzą programiści w procesie programowania.

Zanim zaczniemy programować, musimy sobie zdać sobie sprawę, jaki ma to sens. Wiemy, że komputer znacznie różni się od innych urządzeń codziennego użytku, które mają konkretne zastosowania. Telewizor, na przykład, służy tylko i wyłącznie do oglądania filmów, nie możesz za jego pomocą ugotować sobie zupy, czy przyrządzić kawy; telewizor ma konkretne zastosowanie.

Komputer, natomiast, to coś więcej niż duży, kosztowny kalkulator. Można go wykorzystać do wielu różnych czynności (np. można za jego pomocą słuchać muzyki, oglądać filmy, tworzyć dokumenty, grać, itp.), jednak pod jednym warunkiem: trzeba mieć odpowiedni program. To właśnie od programu zależą zastosowania komputera (np. aby słuchać muzyki, stworzyć dokument, itp. musisz mieć odpowiedni program); komputer bez programu nie robi nic więcej poza zajmowaniem miejsca na biurku.

Program komputerowy jest to algorytm zapisany w języku programowania.

Programowanie to modyfikowanie, rozszerzanie, naprawianie, ale przede wszystkim tworzenie oprogramowania.

Kod źródłowy (również źródło lub źródła) to program komputerowy napisany w jednym z języków programowania (np. Delphi, C++, JavaScript, Java itp.). Kod źródłowy zazwyczaj jest tekstem, który jest zrozumiały dla danego kompilatora języka programowania, jak również dla programisty piszącego program. Kod źródłowy jest przetwarzany przez specjalny program zwany translatorem na kod maszynowy, zrozumiały dla maszyny (procesora).

procesor

(*ang. Central Processing Unit - CPU*) główny układ scalony komputera, wykonuje wszystkie obliczenia i steruje urządzeniami zewnętrznymi.

Translatory

Najważniejszą częścią komputera jest *procesor*. Jest on "mózgiem" komputera i steruje wszystkimi jego elementami. Podstawowym zadaniem procesora jest wykonywanie programu zapisanego w pamięci komputera. Ale procesor „rozumie” program zapisany w „języku procesora”, czyli program zapisany w postaci ciągu instrukcji składających się z samych zer i jedynek.

Zapamiętaj: tylko program zapisany w postaci zer i jedynek jest zrozumiały dla procesora.

Procesor wykonuje program w następujący sposób: pobiera jedną instrukcję z pamięci, wykonuje ją; pobiera następną instrukcję i wykonuje, itd. Instrukcje wykonywane są w kolejności, w jakiej program jest zapisany w pamięci (program jest algorytmem czyli zachowanie kolejności jest niezbędne).

Program komputerowy jest to algorytm zapisany w języku programowania. Warto zauważyć, iż pod terminem język programowania kryje się nie tylko język procesora (czyli zera i jedynki), ale także inne języki bardziej zrozumiałe dla ludzi (np. C, Pascal, PHP itp.). Co zrobić w takim razie żeby procesor zrozumiał programy zapisane w tych innych językach programowania, skoro zrozumiałym dla niego jest tylko język procesora? Należy przetłumaczyć program zapisany w takim języku na język maszynowy (język procesora). Pomysł aby takie tłumaczenie było wykonywane ręcznie przez ludzi jest pozbawiony sensu (lepiej od razu napisać program w języku procesora) dlatego zostały stworzone specjalne programy zwane translatorami.

Translator jest to program tłumaczący programy napisane w określonym języku programowania na język maszynowy. Tłumaczenie takie jest konieczne, ponieważ procesor rozumie tylko język maszynowy.

Każdy program napisany w języku innym niż język procesora musi zostać przetłumaczony przez translator aby komputer mógł go zrozumieć. Mamy dwa rodzaje translatorów:

- interpretery (ang. interpreters)
- kompilatory (ang. compilers)

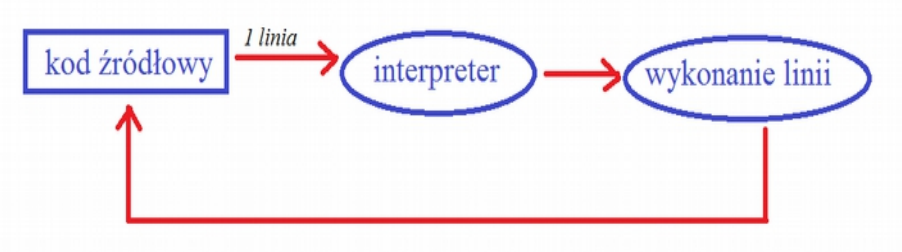
oba różnią się między sobą znacznie pod względem sposobu tłumaczenia programu.

Interpretery

Interpretery tłumaczą program w następujący sposób:

1. pobierają jedną instrukcję programu
2. tłumaczą na język maszynowy
3. dają procesorowi do wykonania
4. powracają do pkt. 1.

Językami interpretowanymi są na przykład: *BASIC, SQL, JavaScript*, itp.



Dobre i złe strony interpreterów

Aby uruchomić program zapisany w języku interpretowanym należy zawsze posiadać interpreter, bez niego program po prostu nie będzie działać. W związku z tym jeżeli chcemy sprzedać komuś program napisany w języku interpretera należy sprzedać go razem z interpreterem. Programy zapisane w języku interpretowanym działają powoli, np. jeżeli w programie jakaś instrukcja ma być wykonywana tysiąc razy to interpreter będzie ją tysiąc razy tłumaczył.

Dobrą stroną interpreterów jest fakt, że programy napisane w ich językach charakteryzują się bardzo dużą przenośnością. To znaczy, że program napisany np. na komputerze PC na 99% będzie działał na komputerach iMac (należy oczywiście mieć odpowiedni interpreter dla iMac-a)

Kompilatory

Kompilatory działają w następujący sposób:

1. tłumaczą cały program
2. zapisują przetłumaczony program w pamięci komputera lub na dysku

Program po przetłumaczeniu (skompilowaniu) przez kompilator jest gotowy do wykonania.

W wyniku kompilacji otrzymujemy kod maszynowy, zwany kodem binarnym.

Najczęściej po kompilacji potrzebna jest jeszcze **konsolidacja**, zwana też popularnie linkowaniem (od terminu angielskiego **link** czyli łączyć, przyłączać). Polega ona na scaleniu binarnych fragmentów programu w jedną całość i dołączeniu procedur systemowych, procedur wejścia/wyjścia, funkcji matematycznych z bibliotek systemowych i innych elementów koniecznych do działania programu.

Językami kompilowanymi są na przykład: *Pascal, C, C++, Asembler, COBOL,* itp.

dobre i złe strony kompilatorów

Programy kompilowane wykonują się szybciej niż interpretowane (nie muszą być na bieżąco tłumaczone ponieważ są już przetłumaczone). Poza tym nie potrzeba kompilatora aby uruchomić program (przetłumaczony przez kompilator program nie musi być tłumaczony przy każdym uruchomieniu).

Złą stroną kompilatorów jest przenośność programów. Programy skompilowane za pomocą kompilatora są w większości nieprzenośne, np. program napisany dla komputera iMac nie będzie działał na komputerze PC. Aby program zadziałał na innym rodzaju komputera należy program na tym komputerze skompilować.

kod wynikowy

(*ang. executable code*) kod wykonywalny. Przetłumaczony przez kompilator *kod źródłowy*, zrozumiały dla komputera, zapisany w postaci ciągu liczb binarnych

Języki programowania - Istnieją setki różnych języków programowania, mniej lub bardziej popularnych. Niektóre z nich prawie w ogóle nie są używane, inne natomiast są tak znane i powszechne, że każdy szanujący się programista powinien znać przynajmniej ich podstawy. Można zadać sobie pytanie, po co istnieje tyle języków programowania. W końcu byłoby znacznie łatwiej, gdyby istniał jeden język, którego używaliby wszyscy programiści. Ewolucja w świecie komputerów wymagała jednak nowych narzędzi — prostszych, szybszych i potężniejszych.
Generalnie języki programowania można podzielić na kilka kategorii, w zależności od stopnia trudności, możliwości czy przeznaczenia. Np. do tworzenia aplikacji na stronach WWW (tzw. skryptów) bardziej nadaje się język PHP, gdyż jest prosty, szybki i zapewnia dużo możliwości. Język Perl, pomimo że również się do tego nadaje, jest trudniejszy, programowanie w nim nie jest tak szybkie i przyjemne, aczkolwiek można go użyć do innych czynności. Język C natomiast daje większą możliwość manipulacji komputerem, jest bardziej zaawansowany. Nie wspominając już o Asemblerze, który daje nam niesamowite możliwości, lecz jest bardzo trudny. Język jest tylko narzędziem w rękach programisty, który musi go odpowiednio wykorzystać, zależnie od sytuacji.

Języki programowania można podzielić ze względu na:

- **wzorzec programowania**
  - •liniowe - BASIC, Fortran
  - •strukturalne - Pascal, C
  - •zdarzeniowe - Visual Basic
  - •obiektowe - C++, Object Pascal, Java
- **generację języków programowania**
  - •języki pierwszej generacji - języki maszynowe, czyli języki procesorów. Instrukcje zapisane są w postaci liczb binarnych.
  - •języki drugiej generacji - języki symboliczne, asemblery. Języki niskiego poziomu, pod względem składni tożsame z maszynowymi, z tą różnicą, że zamiast liczb używa się tu łatwiejszych do zapamiętania mnemonikonów.
  - •języki trzeciej generacji - języki wysokiego poziomu, proceduralne (imperatywne). W tych językach jedna instrukcja jest tłumaczona na kilka instrukcji procesora.
  - •języki czwartej generacji - języki bardzo wysokiego poziomu, nieproceduralne (deklaratywne). Korzystając z tych języków programista skupia się na problemie, a nie na sposobie jego rozwiązania. Semantyka wielu języków tej generacji przypomina składnię języka naturalnego. Przykład - język SQL.
  - •języki piątej generacji - języki sztucznej inteligencji, najbardziej zbliżone do języka naturalnego. Przykład - język PROLOG.
- sposób **kontroli typów**

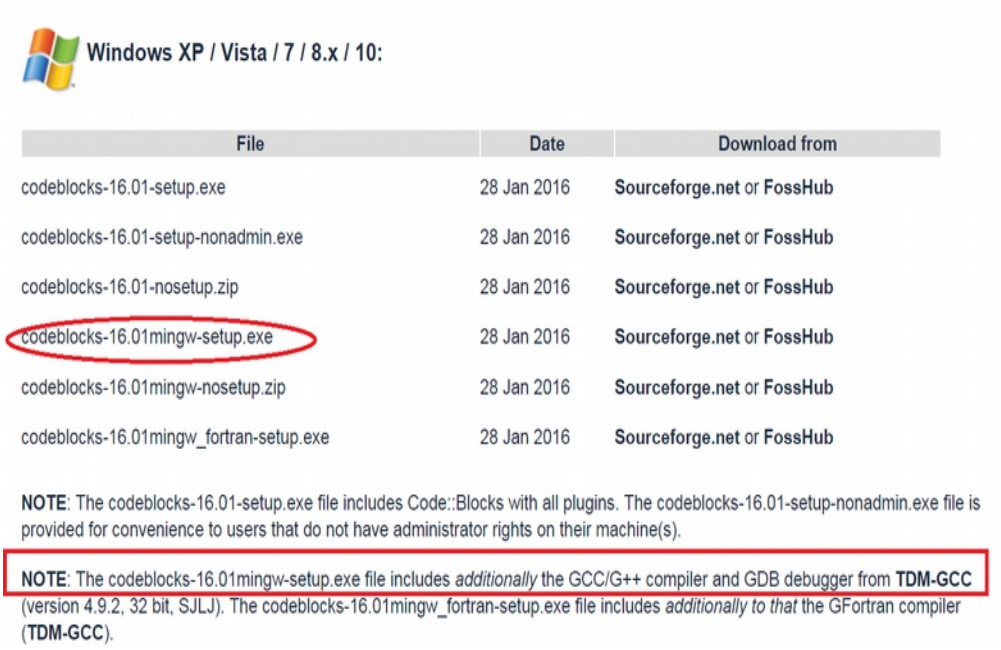
- **sposób wykonania** (kompilacja, interpretacja)
- **poziom** języka programowania
  - •języki **wysokiego poziomu**, np. Pascal, C++
  - •języki **niskiego poziomu** (poziom maszynowy),. np Assemblery
- **przeznaczenie** efektów pracy
  - •tworzenie aplikacji internetowych - Java, JS, PHP
  - •dostęp do baz danych - SQL
  - •obliczenia matematyczne - Fortran
  - •dydaktyczne - LOGO
  - •inne (uniwersalne) - Pascal, C/C++
  - •programowanie wizualne - Visual C, Visual Basic, Delphi
  - •opis danych - PostScript, HTML, XML
  - •tworzenie aplikacji współbieżnych - Ada, Occam
  - •przetwarzanie tekstu - PERL, REXX, Python
  - •programowanie sztucznej inteligencji - LISP, Prolog
  - •programowanie grafiki - OpenGL

Zintegrowane środowisko programistyczne-IDE

Zintegrowane środowisko programistyczne (ang. *Integrated Development Environment*, IDE) – aplikacja lub zespół aplikacji (środowisko) służących do tworzenia, modyfikowania, testowania i konserwacji oprogramowania. Aplikacje będące zintegrowanymi środowiskami programistycznymi charakteryzują się tym, że udostępniają złożoną, wieloraką funkcjonalność obejmującą edycję kodu źródłowego, kompilowanie kodu źródłowego, tworzenie zasobów programu itp. //Wikipedia

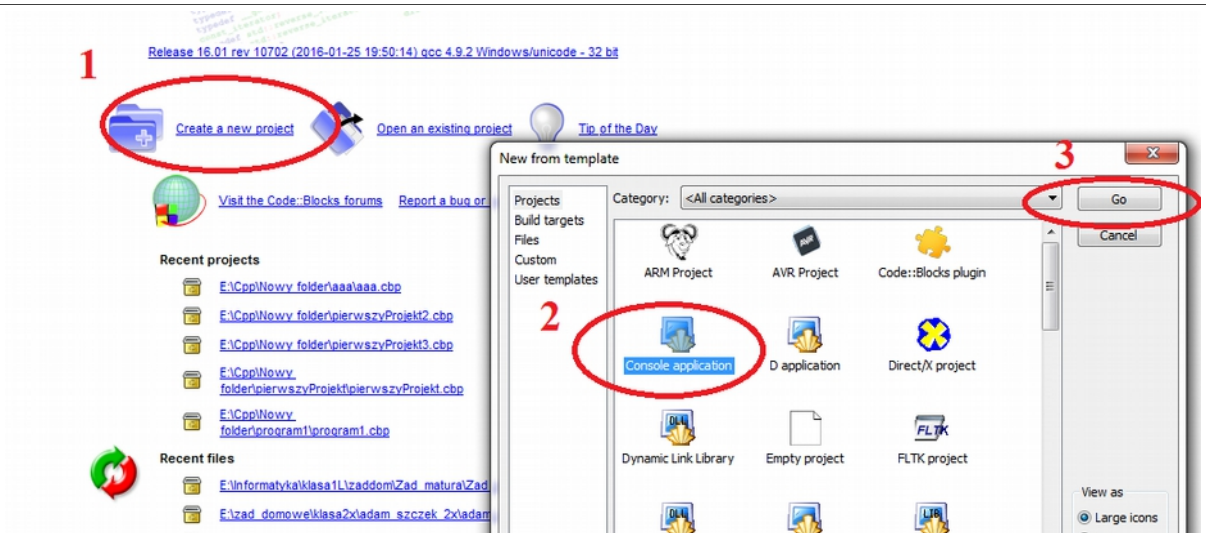
IDE -to pakiet aplikacji ułatwiających tworzenie programów w danym języku programowania. Umożliwia najczęściej organizowanie plików z kodem w projekty, łatwą kompilację, czasem też wizualne tworzenie okien dialogowych. Popularnie, środowisko programistyczne nazywa się po prostu kompilatorem (gdyż jest jego główną częścią).

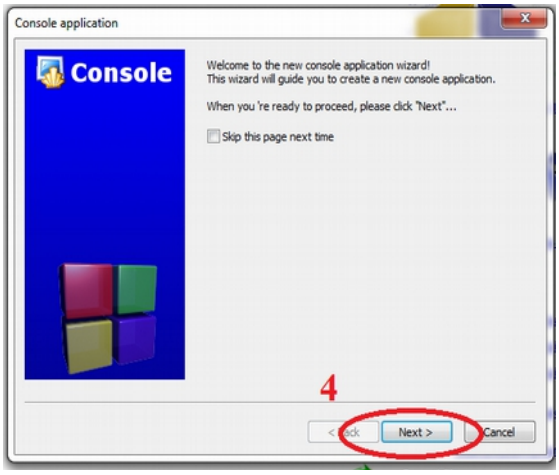
http://www.codeblocks.org/downloads

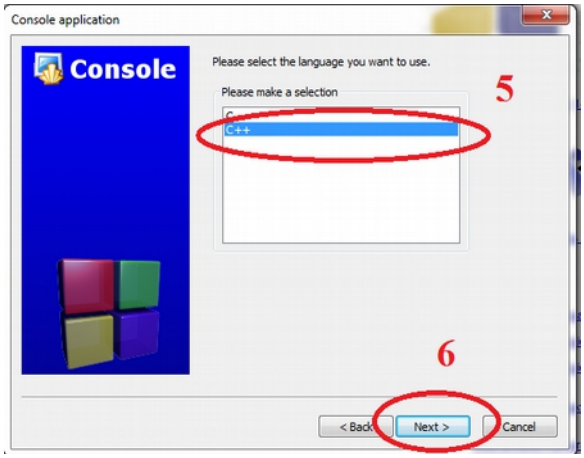


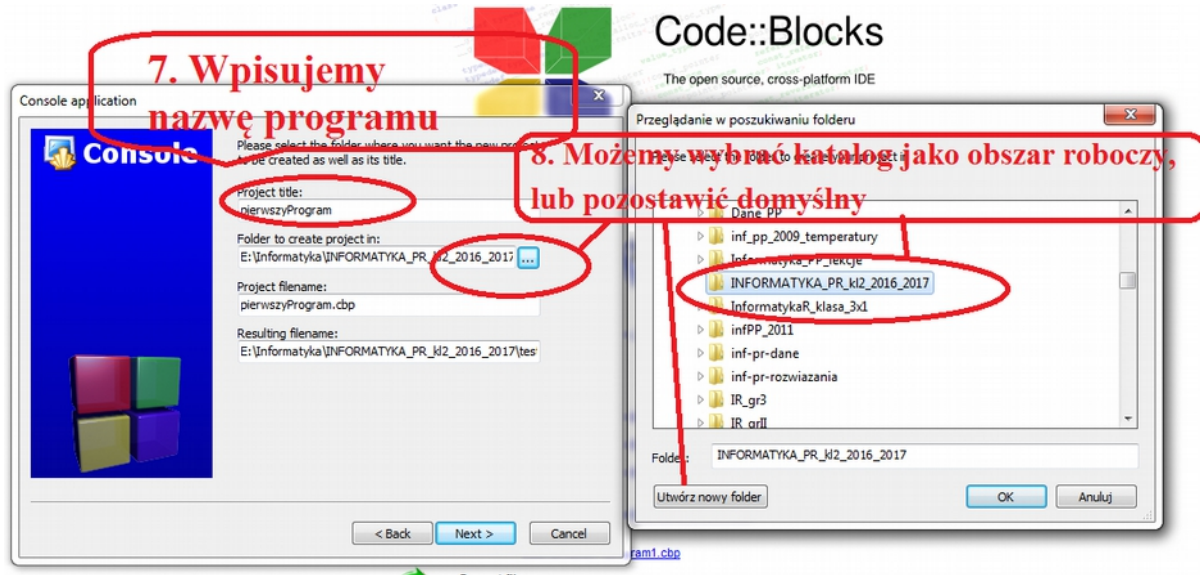
Pozostałe ustawienia domyślne.

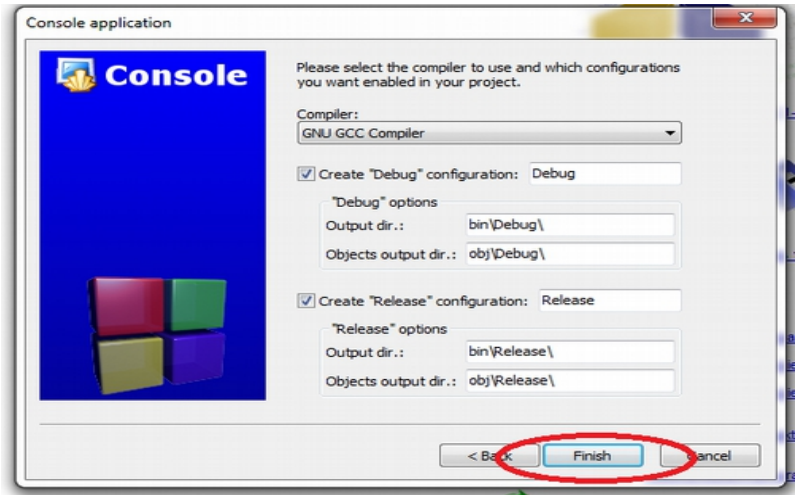
Pierwszy program:

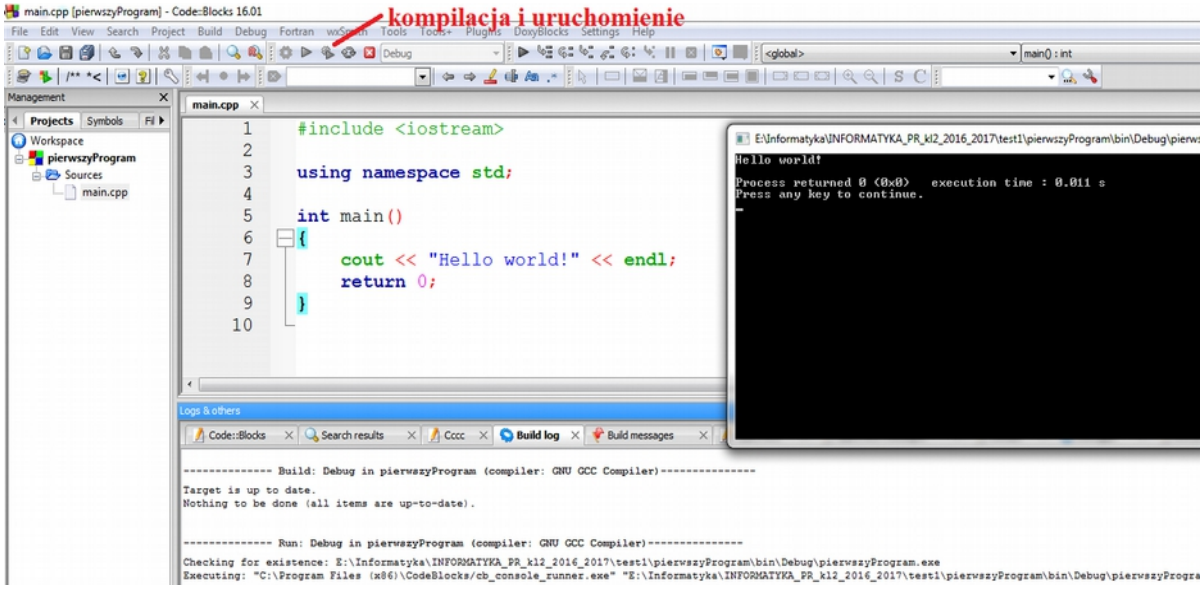

1
Release 16.01 rev 10702 (2016-01-25 19:50:14) gcc 4.9.2 Windows/unicode - 32 bit
Create a new project
Open an existing project
Tip of the Day
Visit the Code::Blocks forums
Report a bug or
Recent projects
E:\Cpp\Nowy folder\aaa\aaa.cbp
E:\Cpp\Nowy folder\pierwszyProjekt2.cbp
E:\Cpp\Nowy folder\pierwszyProjekt3.cbp
E:\Cpp\Nowy folder\pierwszyProjekt\pierwszyProjekt.cbp
E:\Cpp\Nowy folder\program1\program1.cbp
Recent files
E:\Informatyka\klasa1L\zaddom\Zad_matura\Zad
E:\zad_domowe\klasa2x\adam_szczek_2x\adam
New from template
3
Projects
Build targets
Files
Custom
User templates
Category:
<All categories>
Go
Cancel
ARM Project
AVR Project
Code::Blocks plugin
2
Console application
D application
Direct/X project
Dynamic Link Library
Empty project
FLTK project
View as
Large icons


Console application
Console
Welcome to the new console application wizard!
This wizard will guide you to create a new console application.
When you're ready to proceed, please click "Next"...
Skip this page next time
4
< Back
Next >
Cancel

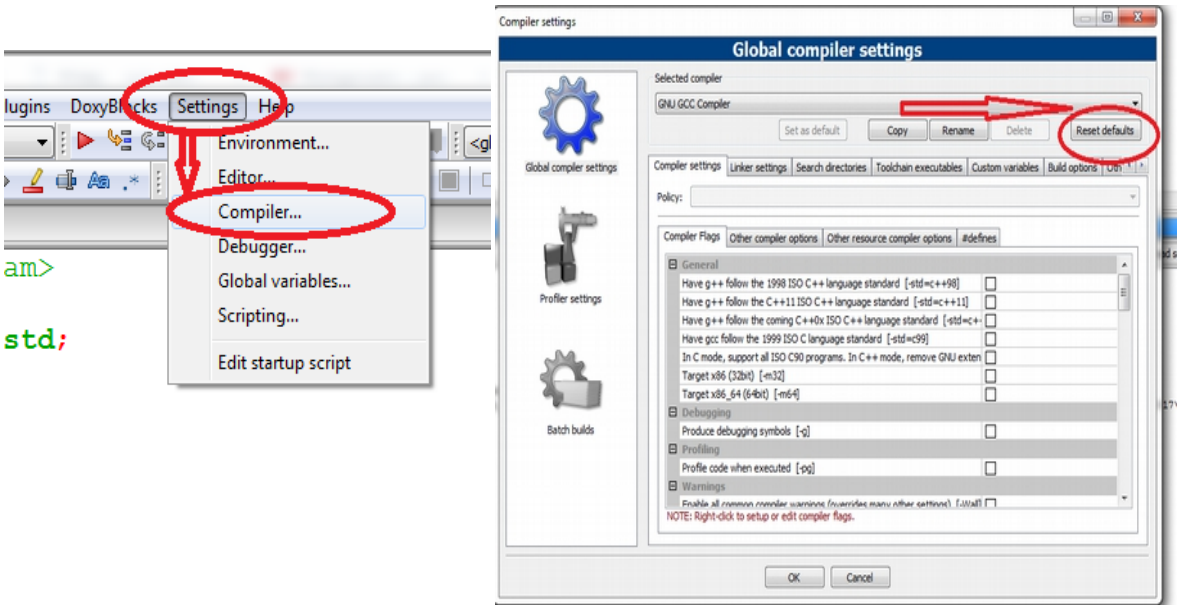

Console application
Console
Please select the language you want to use.
Please make a selection
C++
5
6
< Back
Next >
Cancel


7. Wpisujemy nazwę programu
8. Możemy wybrać katalog jako obszar roboczy, lub pozostawić domyślny
Code::Blocks
The open source, cross-platform IDE
Console application
Console
Project title:
pierwszyProgram
Folder to create project in:
E:\Informatyka\INFORMATYKA_PR_kl2_2016_2017
Project filename:
pierwszyProgram.cbp
Resulting filename:
E:\Informatyka\INFORMATYKA_PR_kl2_2016_2017\tes
< Back
Next >
Cancel
Przeglądanie w poszukiwaniu folderu
Dane_PP
inf_pp_2009_temperatury
INFORMATYKA_PR_kl2_2016_2017
InformatykaR_klasa_3x1
infPP_2011
inf-pr-dane
inf-pr-rozwiazania
IR_gr3
Folder:
INFORMATYKA_PR_kl2_2016_2017
Utwórz nowy folder
OK
Anuluj

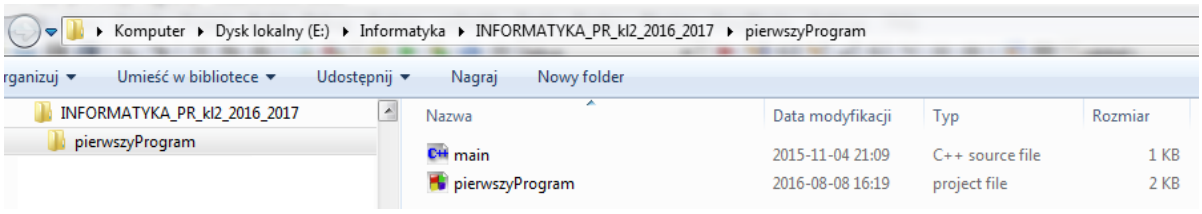




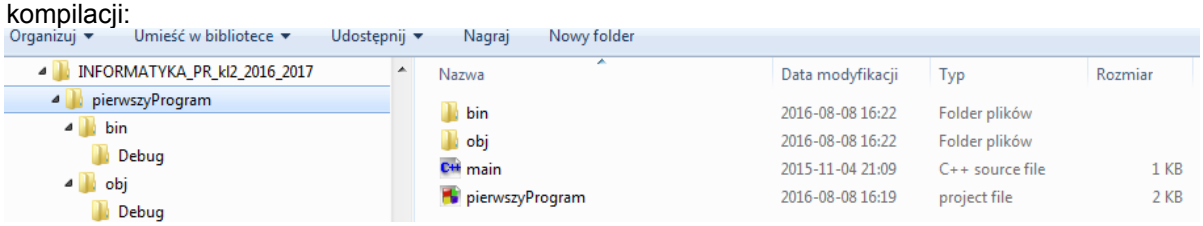
Gdyby pojawiły się problemy z kompilacją, np. kompilator nie został wykryty, to włączamy ustawienia domyślne:



Pliki i katalogi:
a) przed kompilacją:



b) po kompilacji:



Pierwszy program w c++ - analiza pliku źródłowego:

```
1   #include <iostream>
2
3   using namespace std;
4
5   int main()
6   {
7       cout << "Hello world!" << endl;
8       return 0;
9   }
10
```

#include... (linia 1) - włączenie do programu zawartości pliku-biblioteki **iostream**. Dzięki temu w programie dostępne będą narzędzia (w postaci najrozmaitszych klas, funkcji, stałych itd.), służące do wykonywania operacji wejścia/wyjścia (w skrócie, *operacje we/wy*), a więc np. wczytywania z konsoli i wypisywania na ekranie danych. Instrukcja **#include** powoduje rzeczywiste włączenie pliku o nazwie **iostream**, równie dobrze moglibyśmy w tym miejscu wpisać jego treść bezpośrednio do naszego programu.
Sam plik **iostream** znajduje się w znanym kompilatorowi katalogu: użycie nawiasów kątowych ('<...>') oznacza właśnie, że nie jest to nasz własny plik, ale plik ze znanego kompilatorowi specjalnego katalogu bibliotecznego, dostarczonego zwykle wraz z kompilatorem przez producenta. Gdyby chodziło o dołączenie naszego własnego pliku, co też jest możliwe, użylibyśmy podobnej instrukcji, ale zamiast nawiasów kątowych umieścilibyśmy cudzysłowy. Tak więc '**#include <jakasNazwa>**' włącza standardowy plik nagłówkowy o nazwie jakasNazwa (dostarczany zwykle wraz z kompilatorem), natomiast podobna dyrektywa **#include jakasNazwa**" dołączyłaby plik nagłówkowy **jakasNazwa** z katalogu bieżącego, a tylko jeśli w katalogu tym takiego pliku nie byłoby, poszukiwany byłby w katalogu standardowym. Na uwagę zasługuje fakt, że instrukcja '**#include**' *nie* jest przeznaczona dla kompilatora. Włączenie pliku wykonywane jest przez **preprocesor**, który zajmuje się wyłącznie przetwarzaniem **tekstu** naszego programu przed jego właściwą kompilacją - to, co „zobaczy" kompilator, to tekst naszego programu przetworzony przez preprocesor.

Preprocesor jest programem, dokonującym wstępnej obróbki kodu źródłowego przed jego

kompilacją.

using namespace std; (linia 3) - linia ta oznacza, że nazw (klas, stałych, funkcji) niezdefiniowanych w naszym programie należy szukać w przestrzeni nazw *'std'*. W tej przestrzeni nazw znajdują się właśnie obiekty dostarczone przez dyrektywę preprocesora '#include <iostream>': jak widać, samo dołączenie pliku iostream nie wystarcza - obiekty tam zdefiniowane są „zamknięte" w przestrzeni nazw *'std'*. Alternatywnie moglibyśmy dołączyć plik iostream.h (tym razem z rozszerzeniem .h); spowodowałoby to automatyczne „otwarcie" przestrzeni nazw *'std'* (ta forma jest już przestarzała i nie powinna być stosowana w nowych programach) ; instrukcja 'using namespace std;' nie byłaby wtedy potrzebna.

Funkcja **main** (linia 5)-Każdy program języka C++ ma funkcję podstawową (główną), która musi mieć nazwę **main().** Kiedy uruchamiamy nasz program, zaczyna on wykonywać kod zawarty w funkcji main(). Od niej więc rozpoczyna się działanie aplikacji – a nawet więcej: na niej też to działanie się kończy. Zatem program (konsolowy) to przede wszystkim kod zawarty w funkcji main() – determinuje on bezpośrednio jego zachowanie. Funkcja **main** zwraca (instrukcja return) wartość typu **int** (czyli liczbę całkowitą). Instrukcja return 0 informuje, że funkcja main() kończy pracę z wartością 0.Dla większości systemów operacyjnych jest to informacja, o pomyślnym zakończeniu programu.

(linia 7) -Dyrektywa #include<iostream> informuje kompilator, aby pobrał plik „iostream", który zawiera deklaracje strumieni wejścia i wyjścia: cin, cout, operatora wyjścia(<<) i operatora ekstrakcji (>>) Umożliwia to wykonanie poprawnego łączenia kodu wynikowego biblioteki „iostream" z instrukcjami wejscia/wyjscia w programie. Strzałki << i >> oznaczają kierunek, w którym poruszają się dane. Operator << nazywamy operatorem wyjścia (lub operatorem wstawiania), a cout jest identyfikatorem standardowego wyjścia, najczęściej powiązanym z ekranem. Całosć linii (7) możemy nazwać strumieniem danych skierowanym na ekran. Napis z linii (7) jest stałą tekstową – stringiem. Predefiniowany strumień cout jest obiektem – kierującym dane wyjściowe na ekran komputera.

ZADANIA

1	Program, który umożliwia tłumaczenie programu w języku programowania wysokiego poziomu na kod gotowy do wykonania na komputerze, to P / F kompilator. ☐ ☐ BIOS. ☐ ☐ konsolidator. ☐ ☐ Uzupełnienie: **Konsolidator** (*z ang. linker - łącznik*) to program łączący w trakcie procesu konsolidacji pliki obiektów i biblioteki kodów programu tworząc **aplikację wykonywalną**. Bardzo często program taki wchodzi w skład kompilatora, umożliwiając programistom bardzo szybki eksport napisanego programu do aplikacji wykonywalnej.W systemach uniksowych aplikacja taka zwykle nosi nazwę ld. Linker łączy skompilowane moduły kodu i inne pliki w jeden plik wykonywalny, czyli program (w przypadku Windows – plik EXE).

2 Pojęcie algorytmu. Typy danych, zmienne. Podstawowe instrukcje w C++.

Algorytm – jest to skończony, uporządkowany ciąg czynności mających na celu przetworzenie informacji wejściowych na informacje wyjściowe- informacje wejściowe często nazywane są danymi a wyjściowe- wynikami.

inna definicja:

Algorytmem nazywamy uporządkowany zbiór operacji, taki, ze po ich wykonaniu otrzymuje się rozwiązanie dowolnego zadania z określonej klasy zadań.

Jest to bardzo ważne pojęcie. Algorytm jest czymś w rodzaju przepisu albo instrukcji, która „mówi" aplikacji, co ma zrobić gdy napotka taką czy inną sytuację. Dzięki swoim algorytmom programy wiedzą co zrobić po naciśnięciu przycisku myszki, jak zapisać, otworzyć czy wydrukować plik, jak wyświetlić poprawnie stronę WWW, jak odtworzyć utwór w formacie MP3, jak rozpakować archiwum ZIP i ogólnie – jak wykonywać zadania, do których zostały stworzone.

Dla tak sprecyzowanego pojęcia algorytmu możemy podać zestaw cech, jakie powinien on posiadać:

- **ogólność-co oznacza, ze algorytm przeznaczony jest do rozwiązywania określonej klasy zadań, a nie tylko pojedynczego szczególnego przypadku (czyli np. rozwiązywanie równań różniczkowych 2 rzędu, a nie konkretnego równania).**
- **skończoność -możliwość uzyskania rozwiązania problemu w skończonej ilości kroków.**
- **określoność - jednoznaczność wszystkich wykonywanych w nim operacji.**
- **efektywność - przez co rozumiemy czas potrzebny na wykonanie tego algorytmu (najczęściej liczony w umownych jednostkach np. ilości wykonywanych operacji w zależności od danych wejściowych) lub zapotrzebowanie algorytmu na pamięć.**

Algorytmy pojawiły się dużo wcześniej niż pierwsze maszyny liczące albo komputery z możliwościami realizacji algorytmów w postaci programów komputerowych.

Pierwsze algorytmy tworzyli głównie matematycy. To im przede wszystkim były potrzebne zaplanowane działania w celu wykonywania skomplikowanych, jak na tamte czasy, obliczeń rachunkowych. Nieznane było jeszcze nawet słowo *algorytm*. Słowo *algorytm* wywodzi się od arabskiego przydomka *al-chorezmi* („urodzony w Chorezmie"), noszonego przez matematyka, który nazywał się Muḥammad ibn Mūsā al-Khwārizmī i który żył i pracował w IX w. w Bagdadzie.

Słowo algorytm jest zniekształconym brzmieniem jego nazwiska.

Muḥammad ibn Mūsā al-Khwārizmī (ok. 790 – ok. 840 r. n.e.) był wybitnym matematykiem, astronomem i geografem działającym w kręgu kultury arabskiej. Uczony ten zapoczątkował i rozwinął wiele działów matematyki. Wyjaśnił znaczenie używania zera i stosowania systemu pozycyjnego zapisu liczb. Dzięki jego traktatom dziesiętny system zapisu liczb rozprzestrzenił się w kręgu kultury arabskiej, a później (w XII w.) przeniknął do kultury europejskiej.

Algorytm, oprócz definicji operacji, wymaga również definicji danych, nad którymi pracuje. Definicję danych nazywamy specyfikacją algorytmu lub specyfikacją problemu. Specyfikacja składa się zwykle z dwóch podstawowych części:

•Definicja danych wejściowych - określa ona jakie informacje potrzebne są algorytmowi do znalezienia rozwiązania. Np. dla równania liniowego ax + b = c algorytm potrzebuje współczynników a, b i c do znalezienia rozwiązania.
W definicji podajemy również ograniczenia dla danych wejściowych, jeśli takowe istnieją. Np. w powyższym przykładzie liczby a, b i c mogą być dowolne.

•Definicja danych wyjściowych - określa efekt pracy algorytmu, czyli co jest jego wynikiem, co otrzymamy po zastosowaniu algorytmu dla danych wejściowych. Np. dla powyższego równania liniowego ax + b = c, wynikiem będzie wartość x, która po wstawieniu do równania spełnia je lub komunikat o braku rozwiązania .

•Definicja danych pomocniczych - ta część nie jest obowiązkowa, jednakże znacząco ułatwia implementację algorytmu definiując pomocnicze struktury danych, które są niezbędne w trakcie przetwarzania przez algorytm danych wejściowych.

Istnieje co najmniej kilka sposobów zapisu algorytmów. Poniżej zostaną przedstawione najważniejsze z nich.

- **Język naturalny** - najbardziej rozpowszechniony w życiu sposób zapisu algorytmów Jego zaletami są: prostota, brak potrzeby przyswajania specyficznych konstrukcji formalnych, szeroki zasób możliwego do użycia słownictwa; wadami: mała precyzja oraz możliwość błędnej interpretacji (mała ścisłość zapisu), mała zwięzłość.
- **Schematy blokowe** - sformalizowany zapis algorytmów Algorytm prezentowany jest w postaci bloków operacyjnych oraz linii ilustrujących przepływ sterowania. Jest to sposób graficzny. Wadą jest brak (technicznie) możliwości przedstawiania dużych algorytmów
- **Języki formalne**- najczęściej używany w praktyce sposób zapisu algorytmu. Językiem formalnym możne być dowolny z języków programowania lub specjalnie stworzona w tym celu notacja. Zapewnia ścisłość zapisu.
- **Lista kroków**

Symbole graficzne stosowane w schematach blokowych.

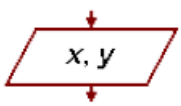
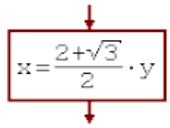
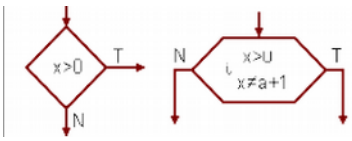
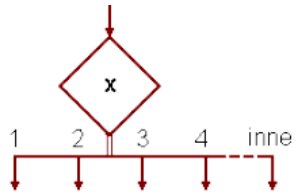
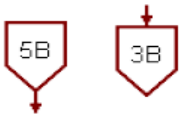
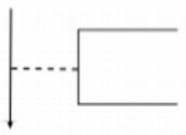
figura geometryczna	opis
	**Blok graniczny-**oznacza on początek, koniec, przerwanie lub wstrzymanie wykonywania działania, np. *blok startu programu*. Początek algorytmu-wewnątrz symbolu wpisuje się zwykle słowo "START" lub nazwę podprogramu. W algorytmie może wystąpić tylko jeden taki symbol. Zakończenie algorytmu - wewnątrz symbolu wpisuje się zwykle słowo "STOP" lub "WRÓĆ", "RETURN" (dla podprogramów). W algorytmie może wystąpić wiele takich symboli.
	**Blok wejścia-wyjścia** – przedstawia czynność wprowadzania danych do programu i przyporządkowania ich zmiennym dla późniejszego wykorzystania, jak i wyprowadzenia wyników obliczeń.
	Bloki wejscia/wyjścia - wygodniej jest jednak odróżnić kierunek wymiany danych. Pierwszy symbol oznacza operacje **wejściowe** (np. dane z klawiatury, z pliku lub parametry wywołania procedury). Drugi symbol oznacza operacje **wyjściowe** (np. na ekran, drukarkę lub do pliku).

figura geometryczna	opis
	**Blok operacyjny** – oznacza wykonanie operacji, w efekcie której zmienią się wartości, postać lub miejsce zapisu danych.
	**Blok decyzyjny, warunkowy** – przedstawia wybór jednego z dwóch wariantów wykonywania programu na podstawie sprawdzenia warunku wpisanego w ów blok
	Odmiana instrukcji warunkowej oznaczająca wybór dalszej ścieżki zależnie od wartości zmiennej lub wyrażenia. Odpowiada ciągowi instrukcji warunkowych ale pozwala na bardziej zwarty zapis algorytmu i programu. (instrukcja switch)
	Łączniki umożliwiają połączenie odległych fragmentów sieci działań. Stosowane są przy dużych sieciach działań. **Łączniki wewnętrzne(wewnątrzstronicowe)** – służą do łączenia odrębnych części schematu znajdujących się na tej samej stronie, powiązane ze sobą łączniki oznaczone są tym samym napisem, np. *A*.
	**Łączniki zewnętrzne (międzystronicowe)** – służą do łączenia odrębnych części schematu znajdujących się na różnych stronach; powinny być opisany jak łączniki wewnętrznne, poza tym powinien zawierać numer strony, do której się odwołuje, np. *4.3, 2,B2*.
	**Blok komentarza** – pozwala wprowadzać komentarze wyjaśniające poszczególne części schematu, co ułatwia zrozumienie go czytającemu, np. *wprowadzenie danych*.
	**Blok wywołania podprogramu** -Wewnątrz zwykle wpisuje się nazwę podprogramu i parametry wywołania.
	blok kolekcyjny - łączy dwie różne drogi algorytmu

Specyfikacja algorytmu– szczegółowy opis zadania w którym wymienia się dane wejściowe i wyniki oraz warunki jakie muszą spełniać, czyli określa się związek między danymi a wynikami.

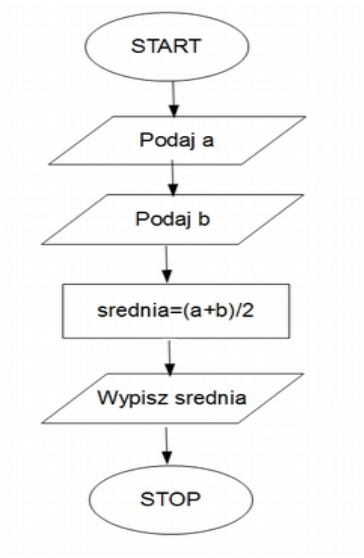
Implementacja to zapis algorytmu w języku programowania.

Algorytmika - podstawowy dział informatyki poświęcony poszukiwaniom, konstruowaniu i badaniom algorytmów, zwłaszcza w kontekście ich przydatności do rozwiązywania problemów za pomocą komputerów.

Zadanie:
Napisz algorytm obliczający średnią arytmetyczną dwóch liczbrzeczywistych. Podaj specyfikacje algorytmu.

Specyfikacja algorytmu:
opis problemu: algorytm obliczający średnią arytmetyczną dwóch liczb rzeczywistych;
opis danych wejściowych: a, b-liczby rzeczywiste;
opis danych wyjściowych: srednia-liczba rzeczywista, srednia=(a+b)/2;

Różne sposoby zapisu algorytmu:

Schemat blokowy	Lista kroków
	Lista kroków charakteryzuje się tym, że każdy wiersz opisujący pojedynczy krok realizowanej czynności jest numerowany. 1) start 2)pobierz liczbę a 3) pobierz liczbę b) 4) dodaj liczby do siebie 5) wynik dodawania podziel przez 2 6) wyświetl otrzymaną wartość 7) zakończ algorytm

Program=algorytm zapisany w języku programowania.

```
#include<iostream>
using namespace std;

int main(){
double a, b, srednia;//definicja zmiennych typu double
cout<<"Podaj liczbe a ";//komunikat dla użytkownika
cin>>a;  //instrukcja wpisujaca do pamieci komputera wartość zmiennej a
/*Strzałki << i >> oznaczają kierunek, w którym poruszają się dane.
Predefiniowane strumienie cout i cin są obiektami
cin - to strumień wejściowy, strumień danych wejściowych przekazywanych do
 komputera
 cout - strumień wyjściowy - to strumień danych  skierowany na ekran  */
cout<<"Podaj liczbe b ";//komunikat dla użytkownika
cin>>b;//instrukcja wpisujaca do pamieci komputera wartość zmiennej b

srednia=(a+b)/2;//zapisanie wyniku do zmiennej srednia
cout<<"Srednia liczb a i b wynosi "<<srednia<<endl;//wypisanie wyniku z komentarzem na ekran

    return 0;
}
```

Powyzszy algorytm jest przykładem algorytmu sekwencyjnego.
Algorytm liniowy (sekwencyjny) – algorytm, w którym kolejność wykonywanych czynności jest taka sama i niezależna od wartości danych wejściowych.

zmienna - to obiekt w programowaniu, który przechowuje różnego rodzaju dane niezbędne do działania programu. Zmienna podczas działania programu może zmieniać swoje wartości (jak wskazuje nazwa). Tworząc zmienną musimy nadać jej nazwę oraz typ, który określa co nasza zmienna będzie przechowywać.

Nadając nazwę (zmiennej, funkcji) trzymamy się następujących reguł:

- nazwa zmiennej jest jednym ciągiem znaków bez spacji np. nazwa_zmiennej - dobrze, nazwa zmiennej - źle
- nie zaczynamy nazwy od cyfry np. 12zmienna - źle, zmienna12 - dobrze
- nie używamy polskich liter takich jak ą, ę itp.
- nazwa zmiennej powinna kojarzyć się z przeznaczeniem tej zmiennej np. tablica_ciagu - dobrze
- nazwa nie może być słowem kluczowym języka programowania np. return - źle

typ zmiennej - tworząc zmienną musimy się zastanowić, jakie będzie jej zastosowanie. Zmienne mogą przechowywać znaki, liczby całkowite, liczby rzeczywiste, ciągi znaków lub wartość logiczną true lub false.

Typy całkowite

Nazwa	Wielkość (bajty)	Zakres
short	2	$-2^{15} \div 2^{15} - 1$, czyli przedział [-32768, 32767]
int	4	$-2^{31} \div 2^{31} - 1$, czyli przedział [-2147483648, 2147483647]
long	4	$-2^{31} \div 2^{31} - 1$, czyli przedział [-2147483648, 2147483647]
long long	8	$-2^{63} \div 2^{63} - 1$, czyli przedział [-9223372036854775808, 9223372036854775807]
unsigned short	2	$0 \div 2^{16} - 1$, czyli przedział [0, 65535]
unsigned int	4	$0 \div 2^{32} - 1$, czyli przedział [0, 4294967295]
unsigned long	4	$0 \div 2^{32} - 1$, czyli przedział [0, 4294967295]
unsigned long long	8	$0 \div 2^{64} - 1$, czyli przedział [0, 18446744073709551615]

Zadanie domowe
1) Podaj algorytmu obliczającego pole prostokąta o bokach długości a i b, podanych przez

użytkownika. Zapisz algorytm w postaci listy kroków, schematu blokowego i programu w języku C++.

Warto zauważyć, że po dodaniu słowa kluczowego unsigned (bez znaku), wartości zmiennych stają się nieujemne i podwojony zostaje prawy zakres.

Typ rzeczywisty - przechowuje liczby zmiennoprzecinkowe.

Nazwa	Wielkość (bajty)	Zakres
float	4	pojedyncza precyzja - dokładność 6 - 7 cyfr po przecinku
double	8	podwójna precyzja - dokładność 15 - 16 cyfr po przecinku
long double	12	liczby z ogromną dokładnością - 19 - 20 cyfr po przecinku

Typ znakowy - przechowuje znaki, które są kodowane kodem ASCII. Tzn. znak w pamięci nie może być przechowany jako znak, tylko jako pewna liczba. Dlatego każdy znak ma swój odpowiednik liczbowy z zakresu [0, 255], który nazywamy kodem ASCII. I na przykład litera "d" ma wartość 100, "!" = 33, itd.:

Typ znakowy

Nazwa	Wielkość (bajty)	Zakres
char	1	-128 ÷ 127
unsigned char	1	0 ÷ 255

Typ logiczny - przechowuje jedną z dwóch wartości - true (prawda) albo false (fałsz). Wartość logiczna true jest równa 1, natomiast false ma wartość 0.

Typ logiczny

Nazwa	Wielkość (bajty)	Wartości
bool	1	true (1) false (0)

Dla zmiennych tego typu możemy realizować przypisanie na dwa sposoby, podając wartość **true** lub **fałsz**, albo 1 lub 0.

Typ string – do przechowywania ciągu znaków.(klasa string jest częścią standardowej biblioteki std)

Etapy rozwiązywania problemów:

1. Sformułowanie zadania
2. Określenie danych wejściowych
3. Określenie celu, czyli wyniku
4. Poszukiwanie metody rozwiązania, czyli algorytmu
5. Przedstawienie algorytmu w postaci:
- opisu słownego
- listy kroków
- schematu blokowego
- jednego z języków programowania
6. Analiza poprawności rozwiązania
7. Testowanie rozwiązania dla różnych danych - ocena efektywności przyjętej metody

1. W języku c++ rozróżnia się małe i wielkie litery (polecenia c++ piszemy małymi literami).
2. Każdą instrukcję w kodzie źródłowym należy kończyć średnikiem.
3. W kodzie programu warto używać komentarzy dla poprawienia czytelności kodu szczególnie w obszernych programach. Komentarz jednowierszowy umieszczamy po znaku //, zaś wielowierszowy między znakami /*komentarz*/.

Zadanie
Iteracja to: (wybierz poprawną odpowiedź)
a) instrukcja zmniejszająca o jeden wartość zmiennej podanej jako argument.
b) wyrażenie matematyczne powodujące zwiększenie wartości zmiennej o jeden.
c) instrukcja pozwalająca na sprawdzenie warunku na poziomie wyrażenia.
ci) d) czynność powtarzania wykonywania instrukcji (ciągu instrukcji) w pętli.