1.Przygotowanie komputera do programowania

Przed rozpoczęciem nauki programowania musimy odpowiednio przygotować komputer. Najpierw należy wybrać kompilator, z którego będziemy korzystali, a potem zainstalować go.

Spośród wielu dostępnych kompilatorów zainstalujemy Dv-C++ ze względu na kilka cech:

►jest całkowicie darmowa, a stworzone za jej pomocą programy można sprzedawać,

►oprócz zwykłego kompilatora jest teżdoskonałym edytorem kolorującym składnię kodu źródłowego,

►dzięki dodatkowym modułom i zestawom narzędzi ułatwia pracę zarówno podczas kodowania, jak również umożliwia tworzenie instalacyjnych wersji programów,

►w aplikacji znajduje się dosyć dobra pomoc (niestety po angielsku), w której opisano samą aplikację oraz język C+ +,

►program wyposażono w kilka ciekawych przykładów różnych kodów źródłowych.

2. Instalacja Dev-C++

Wkładamy do napędu CD płytę i uruchamiamy plik o nazwie devcpp4991setup.

W oknie, które się pojawia, klikamy na przycisk \OK\.

W następnym oknie możemy zapoznać się z warunkami licencji środowiska Dev-C++ 4.

Aby przystąpić do instalacji programu, klikamy na przycisk I agree

Z listy Select the type of instal wybieramy pozycję Fuli (pełną instalację), klikamy na przycisk Next , a w następnym oknie na przycisk install|.

Jeśli pojawia się okno z pytaniem o zainstalowanie aplikacji dla wszystkich użytkowników naszego komputera, klikamy w nim na przycisk Tak

Po chwili aplikacja Dev-C++ zostaje zainstalowana w katalogu C:\Dev-Cpp.

W ostatnim oknie kreatora instalacji klikamy na przycisk Finish - instalacja dobiega końca i Dev-C++ zostaje uruchomiony

Przy pierwszym uruchomieniu aplikacji Dev-C++ należy ją skonfigurować.

Na początku pojawia się okno w którym klikamy na OK.

W następnym oknie możemy określić język aplikacji oraz jej wygląd listy wybieramy pozycję Polish(Polski)

Z listy możemy wybrać motyw (rodzaj wyglądu) aplikacji. Wybieramy New Look.

W trzech kolejnych oknach klikamy na przycisk Next.

Po kilkudziesięciu sekundach proces konfiguracji zostaje zakończony. W oknie, które się pojawia klikamy na przycisk Ok.

3. Poznajemy Dev-C++

Nasz kompilator jest już zainstalowany i skonfigurowany. Dowiemy się teraz, jak go obsługiwać oraz poznamy znaczenie jego najważniejszych elementów i niektórych opcji.

Aby na ekranie pojawiły się wszystkie ważne okna, otwieramy przykładowy projekt.

W tym celu, z menu Plik wybieramy pozycję 0twórz projekt lub plik... .

W oknie, które się otwiera, przechodzimy do katalogu C:\Dev-Cpp\Examples\Hello i dwukrotnie klikamy na ikonę pliku Hello.dey.

Jest to tak zwany plik projektu, w którym znajdują się informacje o nazwie pliku z kodem źródłowym oraz dodatkowe informacje o projekcie.

Projekt zostaje otwarty. Aby podejrzeć jego kod klikamy na jego pozycję w oknie K.

Poznajmy teraz przeznaczenie znajdujących się w aplikacji okien oraz znaczenie najważniejszych ikon.

A -belka tytułowa aplikacji. Oprócz informacji o wersji Dev-C++ znajduje się na niej również informacja o nazwie naszego projektu oraz o nazwie pliku projektu.

B - Menu aplikacji, które umożliwia dostęp do poleceń Dev-C++.

C - Ikona odpowiedzialna za uruchamianie kreatora, pozwalającego utworzyć nowy projekt.

D - Ikona pozwalająca na otwieranie pliku lub projektu.

E -Ikona umożliwiająca zapis wszystkich plików projektu.

F - Ikony pozwalające na przeszukiwanie plików projektu (uruchamiają kolejno: Znajdź tekst, Zamień tekst i Szukaj ponownie)

G - Ikona pozwalająca na przejście do dowolnego wiersza kodu źródłowego -przydaje się przy długich kodach.

H -Ikony pozwalające na dodanie i usunięcie plików z projektu .

I- Ikona opcji projektu.

J - Ikony pozwalające na kompilację i uruchomienie stworzonego przez nas programu.

K - Okno przeglądarki projektu. Znajdują się w niej nazwy wszystkich plików, z których składa się projekt. Po kliknięciu na zakładkę Klasy w oknie tym pojawiają się wszystkie stworzone w ramach projektu klasy, a po kliknięciu na zakładkę Odpluskiwiacz pojawia się okno, w którym można obserwować wartości przypisywane do zmiennych podczas działania aplikacji

L - Zakładki edytora. Dzięki nim możemy w Dev-C++ wygodnie pracować z kilkoma plikami naraz.

M - Edytor kodu źródłowego. W tym oknie widać zawartość pliku, którego nazwa znajduje się na aktywnej zakładce.

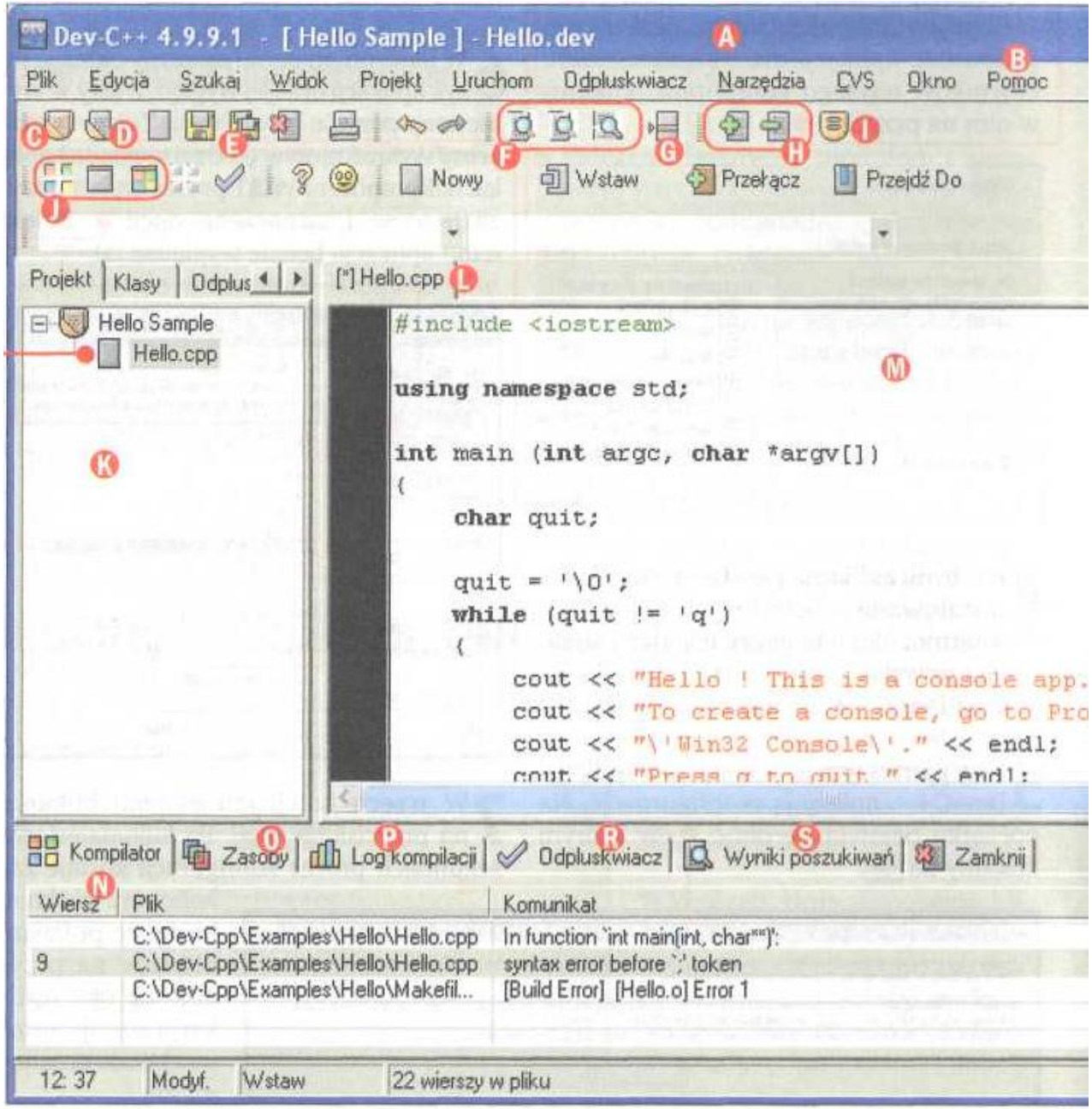
N - Informacje kompilatora. W tym miejscu pojawiają się komunikaty pochodzące od kompilatora (na przykład komunikaty o błędach).

O - Zakładka okna z informacjami dotyczącymi błędów kompilacji, które dotyczą plików zasobów projektu.

P - Zakładka okna z logiem kompilatora.

R - Zakładka okna z opcjami ułatwiającymi poprawianie błędów w naszym programie.

S - Zakładka okna, w którym wyświetlane są wyniki wyszukiwania.


Dev-C++ 4.9.9.1 - [Hello Sample] - Hello.dev
A
Plik Edycja Szukaj Widok Projekt Uruchom Odpluskwiacz Narzędzia CVS Okno Pomoc
B
C
D
E
F
G
H
I
J
Nowy
Wstaw
Przełącz
Przejdź Do
Projekt Klasy Odplus
[*] Hello.cpp
L
Hello Sample
Hello.cpp
K
M
#include <iostream>
using namespace std;
int main (int argc, char *argv[])
{
char quit;
quit = '\0';
while (quit != 'q')
{
cout << "Hello ! This is a console app.
cout << "To create a console, go to Pro
cout << "\'Win32 Console\'." << endl;
Kompilator
O
Zasoby
P
Log kompilacji
R
Odpluskwiacz
S
Wyniki poszukiwań
Zamknij
N
Wiersz
Plik
Komunikat
C:\Dev-Cpp\Examples\Hello\Hello.cpp
In function `int main(int, char**)':
9
C:\Dev-Cpp\Examples\Hello\Hello.cpp
syntax error before `;' token
C:\Dev-Cpp\Examples\Hello\Makefil...
[Build Error] [Hello.o] Error 1
12: 37
Modyf.
Wstaw
22 wierszy w pliku

4. Wydając komputerowi rozkazy, trzeba być bardzo precyzyjnym. Warto więc dobrze zaplanować kod

Programowanie to dość złożony proces. Aby się nie pogubić i jak najszybciej stworzyć poprawnie działającą aplikację, należy odpowiednio wszystko zaplanować.

1.Na początku musimy określić, co chcemy osiągnąć, czyli co nasz program ma wykonywać**1**.. Jest to niezwykle ważna faza programowania i można ją porównać do wyboru posiłku, który mamy zamiar przygotować.

2. Jeśli wiemy już, co chcemy osiągnąć, warto sobie na kartce rozpisać, w jaki sposób program ma działać **2.**. Powinniśmy stworzyć tak zwany algorytm, czyli dokładny opis tego, co program powinien po kolei robić, aby wykonywać określone przez nas zadanie.

3Bardzo dobrym pomysłem jest rozrysowanie naszego algorytmu za pomocą schematu blokowego **3**. Jeśli nabierzemy wprawy, algorytm działania naszego programu możemy od razu tworzyć właśnie w postaci schematu blokowego.

Jeśli dopiero rozpoczynamy naszą przygodę z programowaniem, nie stawiajmy sobie zbyt wygórowanych celów. Lepiej na początku stworzyć program obliczający pole powierzchni jednej figury i go później rozbudowywać, niż od razu porywać się na aplikację rozwiązującą złożone układy równań i wykreślającą dla nich wykresy. Oczywiście, wraz z nabywaniem doświadczenia programistycznego, będziemy potrafili planować i kodować coraz bardziej rozbudowane aplikacje.
Im dokładniejszy schemat wykonamy, tym jego zakodowanie będzie łatwiejsze. Warto więc poświęcić czas na dobre przemyślenie algorytmu naszej aplikacji i przyłożyć się przy tworzeniu schematu blokowego.

4 Teraz możemy już przystąpić do kodowania, czyli przedstawienia schematu blokowego naszego programu za pomocą konkretnych poleceń wybranego języka programowania **4**. To główny etap pracy programisty.

5 Kolejnym etapem jest kompilacja i uruchomienie naszego programu (jeśli kompilacja przebiegła bez błędów) **5**.0.

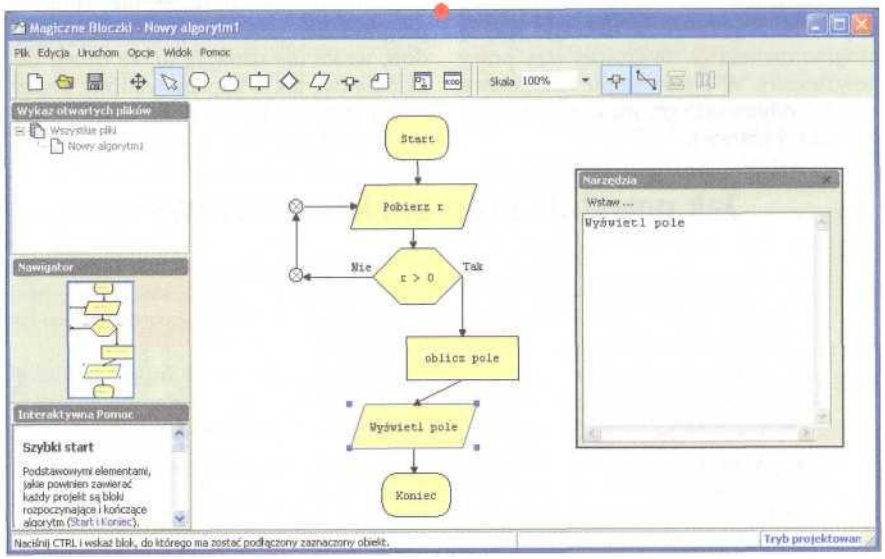
6 Uruchomienie aplikacji to jeszcze nie koniec procesu jej tworzenia. Mimo że program skompilował się wreszcie bez błędów,

wcale nie oznacza to, że działa dokładnie tak, jak to zaplanowaliśmy. Zanim więc pokażemy komuś nasze dzieło, musimy je solidnie przetestować.

5Jak tworzyć schematy blokowe

Schemat blokowy to język służący do opisywania algorytmu działania programu komputerowego w taki sposób, aby na jego podstawie można było napisać kod źródłowy tego programu w niemal dowolnym języku. Umiejętność tworzenia algorytmów i schematów blokowych nie jest więc związana jedynie z językiem C+ +, ale z pewnością przyda się nam w nauce innych języków programowania.

Schematy blokowe najlepiej rysować na kartce papieru. Jednak do ich stworzenia możemy również wykorzystać aplikację Word (w tak zwanych autokształtach znajdują się odpowiednie figury) czy specjalnie do tego celu stworzoną darmową i prostą w obsłudze aplikację Magiczne Bloczki



.

1.Na początku schematu blokowego każdego programu umieszczamy figurę 1 (prostokąt z zaokrąglonymi rogami) oznaczającą początek algorytmu (czyli początek naszego programu).

2 Kolejne bloki naszego schematu zależą już tylko i wyłącznie od wykonywanego przez projektowany program zadania. Jeśli na przykład jego celem będzie obliczenie pola koła na podstawie wprowadzonego przez użytkownika promienia, następną figurą będzie równoległobok. Stosowany jest on do odczytu lub zapisu danych - w tym wypadku do pobrania z klawiatury wartości promienia r 2.

3.Po pobraniu wartości promienia warto sprawdzić, czy wprowadzona przez użytkownika liczba jest większa od zera. Jeśli nie jest, prosimy użytkownika o ponowne wprowadzenie promienia. Gdy wartość jest dodatnia, przechodzimy do dalszej części naszego programu. Blok, który reprezentuje opisaną sytuację (podjęcie decyzji) symbolizowany jest przez romb . Blok ten ma jedną strzałkę wchodzącą i dwie wychodzące (przy jednej wpisujemy słowo TAK - warunek został spełniony, przy drugiej NIE - warunek jest fałszywy).

4. Następnym krokiem jest obliczenie pola koła. Wszystkie obliczenia obrazujemy za pomocą prostokąta 4. Oczywiście we-wnątrz niego nie wpisujemy dokładnych obliczeń, a jedynie krótkie hasło.

5.Kolejną czynnością wykonywaną przez nasz program jest wyświetlenie obliczonego pola na ekranie. Rysujemy więc już znany nam blok wejścia-wyjścia 5 (równoległobok). Wewnątrz niego wpisujemy informację o wyświetleniu danych na ekranie naszego komputera.

6.Każdy stworzony przez nas algorytm powinien skończyć się blokiem z napisem koniec. Oznacza on pomyślne zakończenie działania naszego programu.