PODSTAWY JĘZYKA

C++

CZĘŚĆ I

DANIEL DARUL 2022

Programowanie to proces składający się z etapów:
a)tworzenia programu,
b) testowanie programu.

Język programowania musi mieść określone:
a)Reguły syntaktyczne → reguły **składni** wyrażeń i funkcji użytych w programie,
b)Reguły semantyczne → semantyka języka programowania definiuje precyzyjnie **znaczenie poszczególnych symboli** (instrukcje, operatory itp.) oraz ich funkcję w programie.

Definicja 1.
Algorytm → jest to pewien ciąg czynności, który prowadzi do rozwiązania danego problemu w skończonej ilości kroków.

Definicja 2.
Algorytm → to jednoznaczny przepis, opisujący krok po kroku sposób postępowania w celu rozwiązania pewnego problemu lub sposobu osiągnięcia jakiegoś celu.
Ilość kroków algorytmu zależy od tego, jak złożony jest problem, którego on dotyczy. Zawsze jednak liczba tych kroków będzie **liczbą skończoną**.

Cechy charakterystyczne poprawnego algorytmu:
1. **Poprawność** - dla każdego przypisanego zestawu danych, po wykonaniu skończonej liczby czynności, algorytm prowadzi do poprawnych wyników.
2. **Jednoznaczność** - w każdym przypadku zastosowania algorytmu dla tych samych danych otrzymamy ten sam wynik.
3. **Szczegółowość** - wykonawca algorytmu musi rozumieć opisane czynności i potrafić je wykonywać.
4. **Uniwersalność** - algorytm ma służyć rozwiązywaniu pewnej grupy zadań, a nie tylko jednego zadania. Przykładowo algorytm na rozwiązywanie równań w postaci $ax + b=0$ ma je rozwiązać dla dowolnych współczynników a i b, a nie tylko dla jednego konkretnego zadania, np. $2x + 6 = 0$
5. **Skończoność** – dla każdego zbioru poprawnych danych wejściowych algorytm powinien zwracać wyniki w skończonej liczbie kroków.
6. **Efektywność** – algorytm powinien rozwiązywać problem w jak najmniejszej liczbie kroków.

Etapy konstruowania algorytmu(programu):
1. Sformułowanie zadania.
2. Określenie danych wejściowych.
3. Określenie wyniku oraz sposobu jego prezentacji.
4. Ustalenie metody wykonania zadania.
5. Przy użyciu wybranej metody następuje zapisanie algorytmu.
6. Dokonujemy analizy poprawności rozwiązania.
7. Testowanie rozwiązania dla różnych danych.
8. Ocena skuteczności tegoż algorytmu.

Algorytmy można przedstawiać m.in. następującymi sposobami:

- słowny opis
- schemat blokowy
- lista kroków
- drzewo algorytmu
- drzewo wyrażeń
- w pseudojęzyk
- w język programowania.

<u>Różne sposoby przedstawiania algorytmów</u>

a)opis słowny

Jest na ogół pierwszym, mało ścisłym opisem sposobem rozwiązania problemu. Rozpoczyna się często dyskusją, w jaki sposób można rozwiązać postawione zadanie. Dyskusja służy do rozważań nad sposobem i technikami przydatnymi w rozwiązania problemu.

np. Opis słowny do algorytmu opisującego funkcję modułu (wartość bezwzględną).

Dla wartości dodatnich argumentu x funkcja przyjmuje wartość x, dla wartości ujemnych argumentu x funkcja przyjmuje wartość –x.

b)schemat blokowy

c)lista kroków

Poszczególne kroki zawierają opis operacji, które mają być wykonane prze algorytm. Mogą w nich również wystąpić polecenia związane ze zmianą kolejności wykonywanych kroków. Kolejność kroków jest wykonywana w kolejności ich opisu z wyjątkiem sytuacji gdy jedno z poleceń w kroku jest przejściem do kroku o podanym numerze. Budowa opisu algorytmu w postaci listy kroków jest następujący:

- tytuł algorytmu
- specyfikacja problemu
- lista kroków
- komentarze ujęte w nawiasy klamrowe

{komentarz} <u>uwaga</u>: Krok 0 może być opuszczony

np. Lista kroków dla funkcji SGN(x) czytaj signum.	$SGN(x) = \begin{cases} -1 & dla \quad x < 0 \\ 0 & dla \quad x = 0 \\ 1 & dla \quad x > 0 \end{cases}$

Algorytm obliczania wartości funkcji SGN(x)

Dane: Dowolna liczba rzeczywista x.

Wynik: Wartość funkcji

Krok 0. Wczytaj wartość danej x

Krok 1. Jeśli x>0, to f(x)=1. Zakończ algorytm.

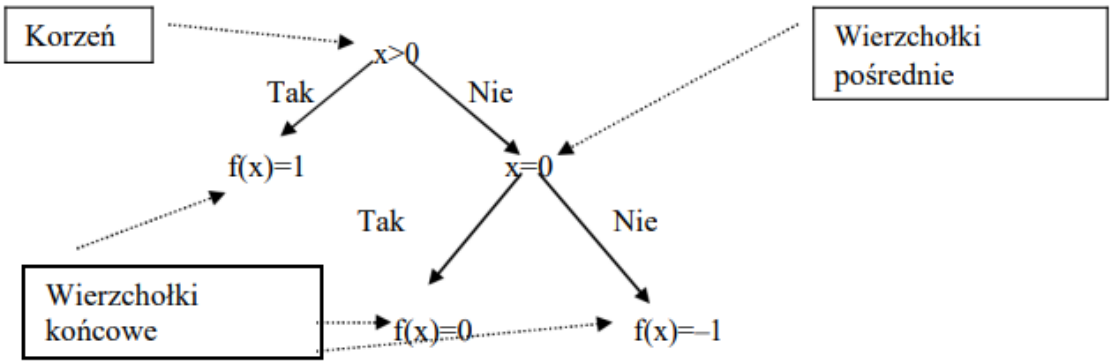
Krok 2. {W tym przypadku x<= 0.} Jeśli x=0, to f(x)=0 . Zakończ algorytm.

Krok 3. {W tym przypadku x< 0.} Mamy f(x)=–1 . Zakończ algorytm.

d)drzewo algorytmu

Nazywany jest również drzewem obliczeń. Każde dwie drogi obliczeń mogą mieć tylko początkowe fragmenty wspólne, ale po rozejściu już się nie spotkają.

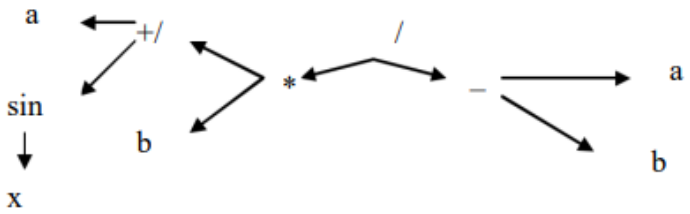
np. Drzewo algorytmu dla funkcji SGN(x).



e)drzewo wyrażeń

Stosowane do obliczeń wyrażeń arytmetycznych.

*np. Wyrażenie (a+sin(x))*b/(a–b)*



f)program w języku programowania np. C++, Pascal

g)pseudojęzyk

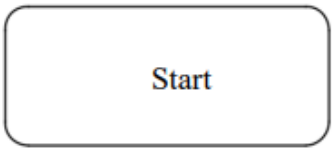
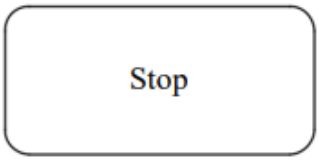
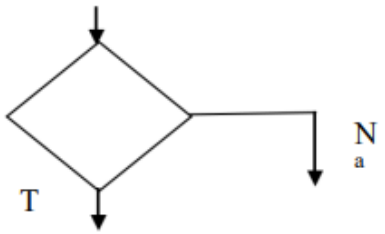
```
PROGRAM Wycieczka;
ZMIENNE punkty:naturalne;
        koszty, dofinansowanie:rzeczywiste;
ZACZNIJ;
        WPROWADŹ(PUNKTY,KOSZTY);
        JEŚLI punkty >=100 i punkty <= TO dofinansowanie :=1/3*koszty+0.2*koszty
                                        W PRZECIWNYM WYPADKU
dofinansowanie:=0.2*koszty;
        WYPROWADŹ('Dofinansowanie wynosi:'dofinansowanie);
ZAKOŃCZ.
```

Specyfikacja problemu

Jest to dokładny opis problemu, który chcemy rozwiązać.

Specyfikacja składa się z:

- **danych wejściowych**,
- **dane wyjściowe** oraz warunki jakie muszą spełniać (czyli związek pomiędzy danymi a wynikami).
- **warunki** jakie muszą spełniać dane wejściowe
- **rysunki** (jeśli są konieczne), **wzory** obliczeniowe

Symbole stosowane w schematach blokowych.		
Początek		W każdym algorytmie musi się znaleźć dokładnie jedna taka figura z napisem "Start" oznaczająca początek algorytmu Blok symbolizujący początek algorytmu ma dokładnie jedną strzałkę wychodzącą
algorytmu Koniec		W każdym algorytmie musi się znaleźć dokładnie jedna figura z napisem "Stop" oznaczająca koniec algorytmu. Najczęściej popełnianym błędem w schematach blokowych jest umieszczanie kilku stanów końcowych, zależnych od sposobu zakończenia programu. Blok symbolizujący koniec ma co najmniej jedną strzałkę wchodzącą.
\wejścia blok wyjścia		Równoległobok jest stosowany do odczytu lub zapisu danych. W jego obrębie należy umieścić stosowną instrukcję np. Read(x) lub Write(x) (można też stosować opis słowny np. "Drukuj x na ekran"). Figura ta ma dokładnie jedną strzałkę wchodzącą i jedną wychodzącą. Jest to blok wyjścia/wejścia wy/we I/O.
mu podprogra blok blok blok przetwarza nia decyzyjny		Romb symbolizuje blok decyzyjny. Umieszcza się w nim jakiś warunek (np. "x>2"). Z dwóch wybranych wierzchołków rombu wyprowadzamy dwie możliwe drogi: gdy warunek jest spełniony (strzałkę wychodzącą z tego wierzchołka należy opatrzyć etykietą "Tak") oraz gdy warunek nie jest spełniony „Niel. Każdy romb ma dokładnie jedną strzałkę wchodzącą oraz dokładnie dwie strzałki wychodzące.
		Jest to figura oznaczająca proces. W jej obrębie umieszczamy wszelkie **obliczenia lub podstawienia**. Proces ma dokładnie jedną strzałkę wchodzącą i dokładnie jedną strzałkę wychodzącą.
		Ta figura symbolizuje proces, który został już kiedyś zdefiniowany. Można ją porównać do procedury, którą definiuje się raz w programie, by następnie móc ją wielokrotnie wywoływać. Warunkiem użycia jest więc wcześniejsze zdefiniowanie procesu. Podobnie jak w przypadku zwykłego procesu i tu mamy jedno wejście i jedno wyjście.

stronicowy łącznik		Koło symbolizuje tzw. łącznik stronicowy. Może się zdarzyć, że chcemy "przeskoczyć" z jednego miejsca na kartce na inne. Możemy w takim wypadku posłużyć się łącznikiem. Umieszczamy w jednym miejscu łącznik z określonym symbolem w środku (np. cyfrą, literą) i doprowadzamy do nic go strzałkę. Następnie w innym miejscu kartki umieszczamy drugi łącznik z takim samym symbolem w środku i wyprowadzamy z niego strzałkę. Łączniki występują więc w parach, jeden ma tylko wejście a drugi wyjście.
łącznik międzystronicowy		Ten symbol to łącznik międzystronicowy. Działa analogicznie jak pierwszy, lecz nie w obrębie strony. Przydatne w złożonych algorytmach, które nie mieszczą się na jednej kartce.
element łączący		Poszczególne elementy schematu łączy się za pomocą strzałek. W większości przypadków blok ma jedną strzałkę wchodzącą i jedną wychodzącą

Reguły rysowania schematów blokowych

I. Po zbudowaniu schematu blokowego nie powinno być takich strzałek, które z nikąd nie wychodzą, lub do nikąd nie dochodzą.

II. Każdy schemat blokowy musi mieć tylko jeden element startowy oraz co najmniej jeden element końca algorytmu.

III. Element łączący(strzałki łączące) powinien być rysowany w poziomie i pionie, załamania pod kątem prostym.

Podział algorytmów.

Definicja algorytmu liniowego

Algorytmem liniowym nazywamy taki algorytm, który ma postać listy kroków wykonywanych **zgodnie z ich kolejnością**.

Algorytmy liniowe są zapisem obliczeń, które mają postać ciągu operacji rachunkowych wykonywanych bez sprawdzania jakichkolwiek warunków.

Algorytm z warunkami (rozgałęzieniami)

Ten typ algorytmu musi mieć bloki decyzyjne czyli bloki sprawdzania warunków.

Algorytm numeryczne

Algorytmy, które wykonują działania matematyczne na danych liczbowych, nazywamy algorytmami numerycznymi.

Algorytm typu dziel i zwyciężaj

Dzielimy problem na kilka mniejszych, a te znowu dzielimy, aż ich rozwiązania staną się oczywiste,

Algorytmy iteracyjne

Iteracja jest to zapętlenie algorytmu, czyli wykonywania danych działań, dopóki warunek iteracji nie zostanie spełniony. Jest ona podstawą wszystkich choć troszkę bardziej złożonych algorytmów. Zazwyczaj ma ona składnię wykonuj "jakaś czynność" dopóki "jakieś wyrażenie logiczne".

Algorytmy rekurencyjne

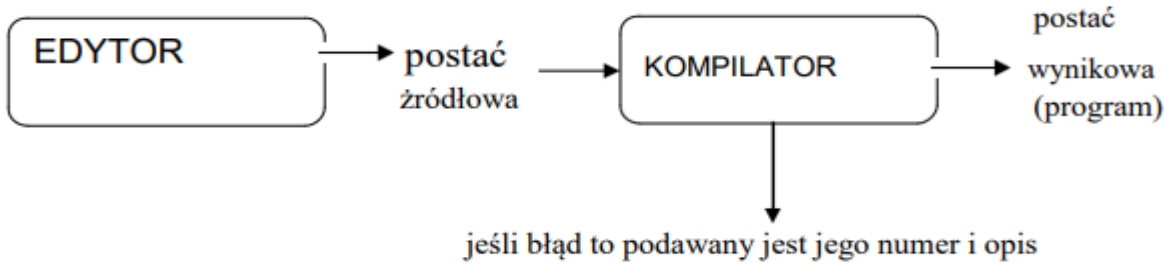
Rekurencje wykorzystuje się do rozwiązywania problemów gdzie powtarza się czynność aby do niego dojść. Swoim działaniem przypomina iteracje. Jednak w tym przypadku **funkcja**

sama siebie wywołuje, dopóki nie otrzyma rozwiązania, natomiast tam mieliśmy powtórzenie pewnej czynności określoną ilość razy.

Złożoność algorytmu- ilość zasobów potrzebnych do poprawnego działania danego algorytmu
Złożoności obliczeniowa-Algorytm wykonujący najmniejszą ilość operacji podstawowych w celu rozwiązania problemu.
Złożoność czasowa- Określa ilość operacji podstawowych potrzebnych do wykonania algorytmu o danej wielkości wejściowej.
Złożoność pamięciowa- Określa ilość przestrzeni pamięci wirtualnej potrzebnej do wykonania algorytmu z określonym zestawem danych wejściowych.

Część 1
Edytor, Kompilator. Tryb tekstowy

System DEV-C++ składa się z:
1. **EDYTORA**– służy do tworzenia tekstu programu w języku C++, traktowanej jako ciąg znaków (litery+liczby+znaki specjalne). Tekst tego programu nazywany jest programem w postaci źródłowej.
2. **KOMPILATORA**–program tłumaczący program w postaci źródłowej na program w postaci kodu asemblera (jest to ciąg operacji maszynowych zakodowanych w postaci liczb dwójkowych). Otrzymujemy plik w pośredni **OBJ**. W celu otrzymania postaci wynikowej konieczne jest użycie **linkera(konsolidatora)**. Linker: łączy pliki *.obj, *.lib, *.dll, generuje plik wykonywalny *.EXE. Postać wynikowa jest gotowa do wykonania jest zapisywany na nośniku w postaci EXE. Gdy program źródłowy ma błędy kompilator sygnalizuje błąd poprzez wskazanie jego miejsca oraz podanie jego numeru.



Tworzenie pierwszego projektu dla trybu DOS.
Plik → Nowy → Projekt → Console Application → Ok

Pojawi się okno gdzie będziemy przechowywać nasz projekt → wybierz miejsce przechowywania i Zapisz
W okienku edycji kodu źródłowego automatycznie jest generowany szkielet aplikacji:

```
#include <cstdlib>      //włączenie modułu użytecznych funkcji
#include <iostream>     //włączenie modułu wejścia-wyjścia

using namespace std;   //zapewnienie użycia funkcji standardowych z biblioteki std

int main(int argc, char *argv[ ]) // początek funkcji
głównej {
   system("PAUSE");              // zatrzymanie programu w okienku DOS
   return EXIT_SUCCESS;      // zwrot wartości funkcji  gdy jest błąd
  }
```

Pliki generowane przez kompilator:
-plik projektu *.dev
-plik kodu źródłowego *.cpp
-plik wykonywalny *.exe
-plik binarny *.o
-plik reguł kompilatora Makefile. win

Kompilowanie programu czyli tłumaczenie z języka CPP na język maszynowy wraz z wykrywanie błędów.
Uruchom → Kompiluj lub Ctrl+F9

Uruchomienie programu gdy kompilacja nie miała błędów to Uruchom → Uruchom lub Ctrl+F10

Polskie znaki na konsoli:

- W treści programu wpisz instrukcję system(lchcp 1250l);
- uruchom Właściwości okna CMD (poprzez kliknięcie granatowego tytułu okna) i w zakładce Czcionki należy wybrać czcionkę Lucida Console i zatwierdzić dolną opcją.

Pytanie 1
Narysuj oraz opisz schemat współpracy edytora z kompilatora.
Pytanie 2
Co to jest postać źródłowa programu.
Pytanie 3
Co to jest kompilator? Jakie wykonuje czynności.
Pytanie 4
Co to jest kod asemblera?
Pytanie 5
Jakie pliki mają rozszerzenia OBJ.?
Pytanie 6
Jakie czynności wykonuje linker(konsolidator)?
Pytanie 7
Jakie rozszerzenie ma plik wykonywalny?
Pytanie 8
b)Zapisz w jaki sposób tworzymy nowy projekt dla trybu DOS.
Pytanie 9
c)Zapisz szkielet aplikacji automatycznie generowanej przez system.
Pytanie 10
Wypisz oraz opisz pliki generowane przez system CPP Dev.
Pytanie 11
Zapisz w jaki sposób kompilujemy programy w DEV
Pytanie 12
Zapisz w jaki sposób uruchamiamy programy w DEV
Pytanie 13
Co to jest komentarz, rodzaje komentarzy z przykładem, jakim kolorem zaznaczany jest komentarz w Dev. **Pytanie 14**
Co to są pliki nagłówkowe. Po jakim słowie kluczowy są pisane pliki nagłówkowe i jak się kończy deklaracja. Znaczenie znaku #. Jakim kolorem oznaczane są dyrektywy preprocesora.
Pytanie 15
Co to jest funkcja główna. Przepisz przykład. Co oznaczają puste nawiasy okrągłe? Co oznaczają nawiasy klamrowe? Znaczenie int.
Pytanie 16
Opisz instrukcję cout
Pytanie 17
w jaki sposób realizujemy przejście do nowej linii z użyciem instrukcji cout
Pytanie 18
omów użycie znaków: # { } ;
Pytanie 19
Opisz polecenie using namespace std;
Pytanie 20
Opisz w jaki sposób używamy instrukcji dosowych. Zapisz cztery przykłady takich funkcji.
Pytanie 21
zapisz uwagi na temat stosowania dużych i małych liter.
Pytanie 22
zapisz uwagi na temat stylu programowania.

Komentarz

Komentarze są to napisy, których kompilator nie bierze pod uwagę podczas kompilacji programu.

a)Komentarz wielolinijkowy.

/* treść komentarza */

b)Komentarz jednoliniowy

Komentarz zaczynający się od // treści komentarza.

Plik nagłówkowy

Każdy program w C++ musi najpierw załączać odpowiednie pliki nagłówkowe z definicjami interesujących nas funkcji (można pisać własne pliki nagłówkowe). Po tych linijkach ze słowem #include (możemy załączać kilka plików, ale wtedy każdemu poświęcamy oddzielną instrukcję #include). Zauważcie, że po instrukcji #include nie ma średnika. Jest ich kilkadziesiąt, a w każdym jest zdefiniowane ok. 50 funkcji.

Opis plików nagłówkowych..

CONIO.H - CONsole Input/Output.

Plik nagłówkowy zawierający prototypy funkcji potrzebnych do obsługi standardowego Wejścia/Wyjścia na/z konsoli (CONsole). Plik zawiera między innymi prototyp funkcji clrscr(), potrzebnej nam do czyszczenia ekranu.

STDIO.H - STanDard Input/Output

Plik nagłówkowy zawierający prototypy funkcji potrzebnych do obsługi standardowego Wejścia/Wyjścia na/z konsoli (Input - Wejście, Output - Wyjście). Plik zawiera między innymi prototyp funkcji **printf()**, potrzebnej nam do drukowania wyników na ekranie.

IOSTREAM.H

Dołącza plik nagłówkowy udostępniający operacje we/wy w stylu C++

Instrukcja **cout** wyprowadza danych (wartości zmiennych, tekstów) z programu z użycie << na ekran (może być też do pliku). Jeśli jest to tekst to ujęty jest on w znaki " ".

cout<<endl; oznacza przejście do nowej linii i może być używane do wykonania pustej linii.

Funkcja główna

Każdy program w C++ musi posiadać funkcję o nazwie **main**. Po słowie main występują nawiasy okrągłe. Puste nawiasy oznaczają, że funkcja nie przyjmuje żadnych argumentów. A sama funkcja nazywa się main (z angielskiego: główna), ponieważ to właśnie instrukcje umieszczone wewnątrz niej (w jej ciele) będą wykonywane przez program. Początek i koniec określania ciała funkcji oznaczamy odpowiednio: otworzonym i zamkniętym nawiasem klamrowym. int oznacza typ wyniku zwracanego przez funkcję główną w tym przypadku jest to liczba całkowita. Dev generuje automatycznie funkcję główną w postaci int main(int argc, char *argv[]). W nawiasie okrągłym są argumenty wejściowe funkcji.

np.

```
int main( )
{

treść programu

}
```

Polecenie using namespace std;

Polecenie rozwiązuje problem dublowania się nazw różnych funkcji i poleceń. Std oznacza bibliotekę standardową , w której znajdują się definicje wszystkich najważniejszych symboli oraz poleceń i funkcji.

Znaki specjalne

znak #

występuje przed dyrektywami proprocesora.

średnik ;

stawiany jest na zakończenie polecenia(instrukcji)

nawiasy klamrowe { }

umożliwiają tworzenie bloku funkcji, czyli łączyć wiele instrukcji prostych tak, aby wyły wykonywane razem.

Użycie instrukcji dosowych

a)dołącz
#include <iostream>
#include <cstdlib>
b)wpisz instrukcję → pamiętaj, że instrukcja dosowa jest w cudzysłowu
system("dir"); lub
system("cls"); →czyszczenie ekranu
system("pause"); →zatrzymanie programu
która zatrzyma program i wyświetli komunikat mniej więcej w tym stylu: **Aby kontynuować. naciśni dowolny klawisz...** .
system("color 5"); →ustawienie koloru tła

Ustawia domyślne kolory tla i pierwszego planu.

COLOR [atr]

atr Określa atrybut koloru dla wyjścia konsoli

Atrybuty kolorów są określone przez DWIE cyfry heksadecymalne
-- pierwsza oznacza tło,
--druga pierwszy plan.
Każda cyfra może być jedną z wartości:

0= Czarny	8 = Szary
1= Niebieski	9 = Jasnoniebieski
2= Zielony	A = Jasnozielony
3= Błękitny	B = Jasnobłękitny
4= Czerwony	C = Jasnoczerwony
5= Purpurowy	D = Jasnopurpurowy
6= żółty	E = Jasnożółty
7= Biały	F = Jaskrawobiały

Jeśli nie podano argumentu, używany jest kolor odpowiadający chwili uruchomienia CMD.EXE. Wartość ta jest brana z bieżącego okna konsoli, z opcji /T wiersza polecenia lub z wartości rejestru DefaultColor.

Polecenie COLOR ustawia ERRORLEVEL na 1, jeśli podjęto próbę określenia
tej samej wartości dla tła i dla pierwszego planu w poleceniu COLOR.

Przykład: "COLOR fc" daje kolor jasnoczerwony na jaskrawobiałym tle.

Uwagi:

- Pamiętaj, że dla języka C++ PRINTF i printf to nie to samo! Słowa kluczowe i nazwy standardowych funkcji **muszą być pisane małymi literami !!!**

Styl programowania

- Można funkcję main zapisać w takiej postaci, jak to zrobiliśmy w naszym programie (czyli w czterech linijkach: nazwa main, nawias, jedna pusta linijka i nawias). Ale można też to wszystko zapisać w jednej linijce, np. main(){ }. Taki sposób nie jest zalecany ze względu na czytelność programu.

- Poza nielicznymi wyjątkami (nie można przedzielać tekstów ujętych w cudzysłów) w języku C++ możemy zrobić przerwę gdzie tylko chcemy.

- void - pusty, wolny

Strumienie

Strumień to przepływ informacji od jednego urządzenia do innego. Strumieniem może być przepływ danych np. tekstu lub wartości zmiennych z pamięci komputera na ekran monitora. Trójkątne nawiasy (<< lub >>) wskazują

kierunek przepływu informacji.

Obiekty

Obiekt podobnie jak program komputerowy jest to grupa danych i funkcji działających wspólnie i przeznaczonych razem do wykonania jakichś zadań. Dla przykładu obiekt **cout** służy do obsługi przesyłania danych na ekran monitora.

Przykład 1

Temat:

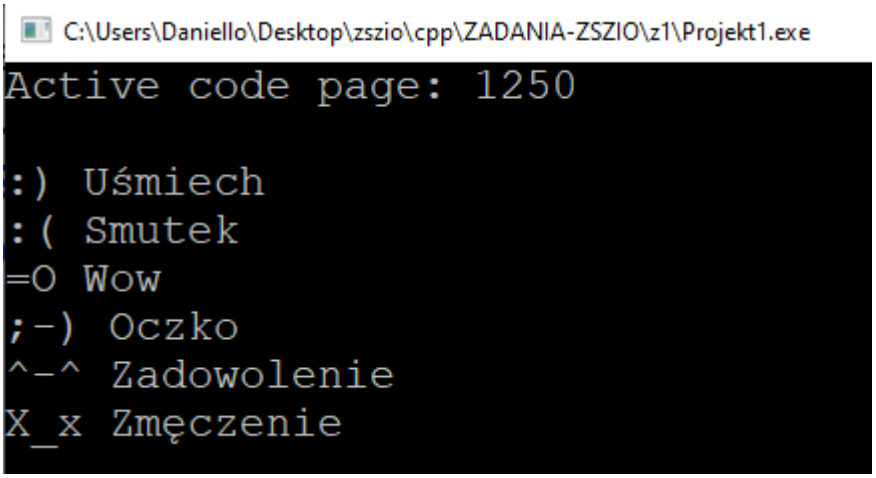
Przykład demonstruj:

- dołączanie plików nagłówkowych,
- wyprowadzanie tekstów na ekran,
- użycie komentarz,
- wprowadzenie pustej linii ekranu,
- zatrzymanie programu.

- Wpisz temat do zeszytu
- Wpisać Przykła1 (łącznie z komentarzem) do komputera. Nazwa pliku na dysku p1_cztery_pierwsze_litery_nazwiska
- Przepisać treści programu do zeszytu.

```
//Początek przykładu1
/* Oto przykład program pokazujący wyprowadzanie tekstu na ekranu sposób1 z użyciem obiektu
cout oraz zatrzymanie działania programu */


#include <iostream>
#include <stdlib.h>
using namespace std;

int main(int argc, char *argv[])
{
  cout<<"dzień dobry, tutaj Twój komputer";
  cout<<endl;
  cout<<endl;
  system("PAUSE");
  return 0;
}
```

Zadanie 1 **Wykonaj program piszący 6 emotikonów, każdy w nowej linii.**



Przykład 2

Temat:

Przykład demonstruj:

- wyprowadzanie tekstów na ekran z użyciem instrukcji Printf,

```
#include <iostream>
#include <stdlib.h>
#include <conio.h>
using namespace std;

int main(int argc, char *argv[])
{


    printf("Autor: ……");
    cout<<endl;
    printf("TO JA, TWOJ PROGRAM-PIERWSZY.CPP");
    cout<<endl;
    system("PAUSE");
    return 0;
  }
```

Przykład 3

Temat: Określanie kolorów tła i liter z użyciem <windows.h>

```
#include <iostream>
#include <windows.h>

using namespace std;

void gotoxy(int x, int y)
{
  COORD coord;
  coord.X = x;
  coord.Y = y;
  SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), coord);
}

int main()
{
  cout << "Tekst nie kolorowany\n\n";
  SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), FOREGROUND_GREEN |
FOREGROUND_INTENSITY);
  cout << "Tekst pokolorowany\n\n";
  SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 15 | FOREGROUND_INTENSITY);
  // 15 to kolor bialy
  cout << "Tekst nie kolorowany\n\n";

  system("pause");
}
```

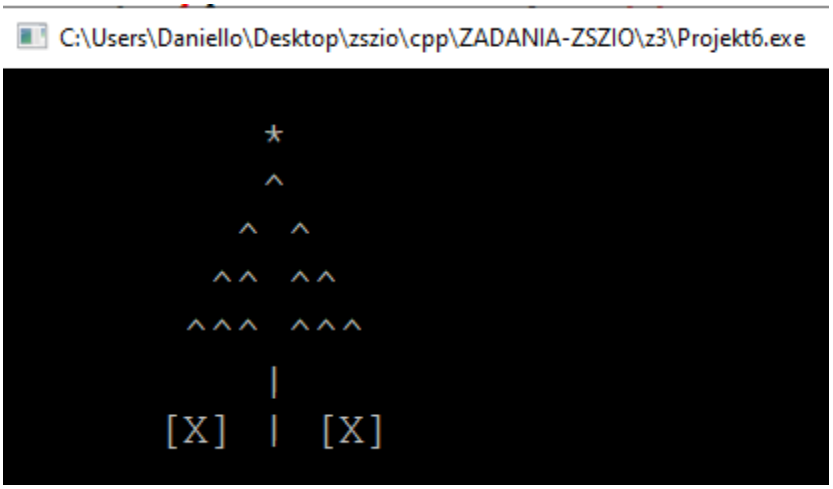
Przykład 3a
Temat: Określanie kolorów tła i liter z użyciem <windows.h>

```
#include <iostream>
#include <windows.h>

using namespace std;
  void gotoxy(int x, int y)
    {
    COORD coord;
    coord.X = x;
    coord.Y = y;
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), coord);
    }
int main()
  {
    gotoxy(10,10);
    cout<<"***";
    gotoxy(20,20);
    cout<<"***";
    system("pause");
  }
```

Przykład 4
Temat: Kolory na konsoli.

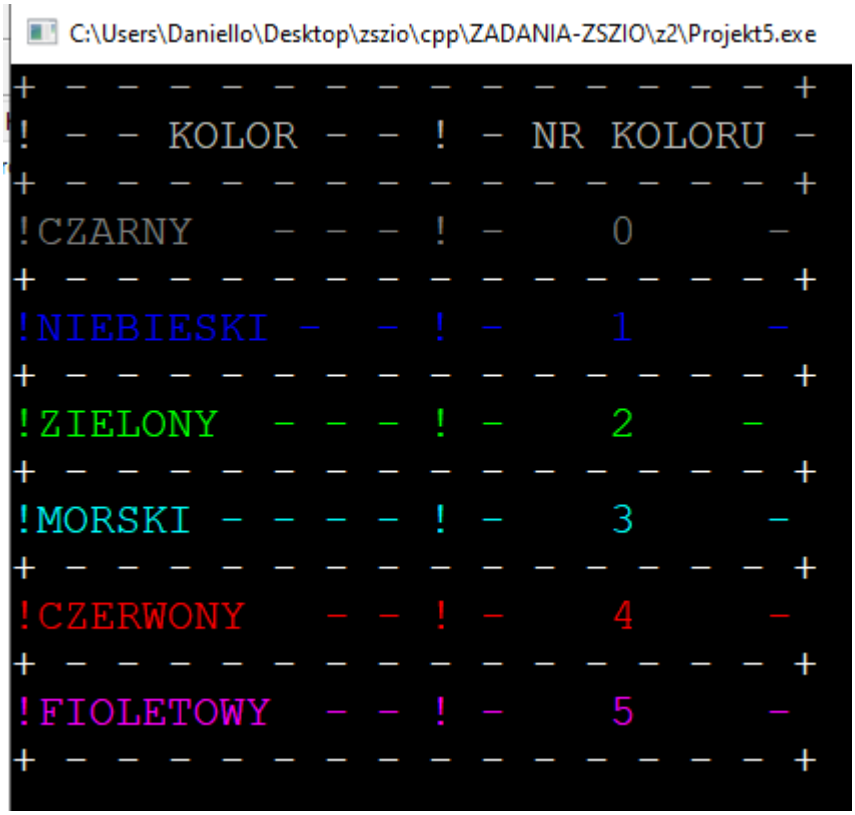
```
#include <windows.h>
#include <iostream>
#include <stdlib.h>

using namespace std;

int main()
{
HANDLE hOut;
hOut = GetStdHandle(STD_OUTPUT_HANDLE);
cout << "Standart" << endl << endl;

SetConsoleTextAttribute(hOut,BACKGROUND_RED);
cout << "czerwony." << flush << endl << endl;

SetConsoleTextAttribute(hOut,FOREGROUND_GREEN);
cout << "zielony." << endl << endl;

SetConsoleTextAttribute(hOut,FOREGROUND_BLUE);
cout << "niebieski." << endl << endl;

system("PAUSE");
return 0;
}
```

Zadanie 2 **Wykonaj program który narysuje choinkę.**



Zadanie 3 **Wykonaj program który narysuje następującą tabelę**

(Do wykonania zadania skorzystaj z przykładów 3,3a,4):



Część 2
Zmienne, stałe, obliczenia, formatowanie wydruków.

Pytanie 1
Omów, jakie nazwy mogą(nie mogą) mieć zmienne i stałe.
Pytanie 2
wymień oraz, krótko opisz sześć podstawowych typów danych wraz z rozmiarem w bajtach..
Pytanie 3
wymień uwagi na temat stosowania nazw,(wypisz 10 słów kluczowych, które nie mogą być nazwami)
Pytanie 4
Omów:

- deklarację,
- inicjację,
- definicje zmiennych

Pytanie 5
narysuj schemat ogólnego podziału danych
Pytanie 6
omów podstawowe modyfikacje do podstawowych danych,
Pytanie 7
omów deklarację stałych na przykładzie
Pytanie 8

a) Omów sposoby zapisu liczb zmiennoprzecinkowych wraz z przydałem.
b) Zamień liczbę z postaci wykładniczej na numer_w_dzienniku**.**miesiąc_urodzeniae4 na liczbę bez potęgi.(tam jest kropka)

Pytanie 9
W jakim miejscu programu można dokonywać deklaracji zmiennych w C++..
Pytanie 10
Omów instrukcję cin. Podaj przykład jednej instrukcji cin do wczytania dwóch zmiennych.
Pytanie 11
Omów instrukcję przypisania w C++.
Pytanie 12
Zapisz co oznacza zapis w treści programu **\n**
Pytanie 13
Opisz Instrukcje **cin** wraz z przykładem oraz przykład wczytania dwóch zmiennych o nazwach **dwie_pierwsze litery_nazwiska** i **dwie_pierwsze_litery_imienia**.
Pytanie 14
Opisz inkrementację oraz dekrementację wraz z przykładem.
Pytanie 15
Przerysuj tabelę operatorów arytmetycznych języka C++.
Pytanie 16
Przerysuj tabelę Operatory relacji języka C++.
Pytanie 17
Zapisz Operatory logiczne języka C++ wraz z przykładem.
Pytanie 18
Omów funkcję **scanf( )**(Znaczenie, składnia bez opisu, przykład1 z opisem, przykład 2 z opisem)
Pytanie 19
Omów funkcję **printf( )** (Znaczenie, składnia bez opisu, przykład1 z opisem, przykład 2 z opisem)
Pytanie 20
Zanotuj trzy wzorce konwersji dla postać %s, %d, %f
Pytanie 21
Opisz trzy zakresy ważności zmiennych.
Pytanie 22
Opisz przykrywanie zmiennych.

Nazwy zmiennych i stałych

Nazwy zmiennych i stałych mogą składać się z liter, cyfr i podkreślenia _ .
Nazwa nie może się zaczynać od cyfry. np. 1promien

Uwagi dotyczące stosowania nazw:

- nazwa powinna być znacząca np. **promien** a nie nic znacząca nazwa **mien,**
- nazwa może się składać z dużych i małych liter lecz musimy pamiętać, że kompilator rozróżnia duże i małe litery, czyli zmienne o nazwach R i r są innymi zmiennymi,
- przyjęto, że nazwy zmiennych piszemy małymi literami a stałych dużymi,
- nie można stosować nazw, które są zarezerwowane dla słów kluczowych:

asm , auto, break, case, char, class, const, continue, default, delete, do, double, else, enum, extern, float, for, friend, goto, huge, if, inline, int, interrupt, long, near, new, operator - operator, pascal, private, protected, public, register, return , short, signed, unsigned , sizeof, static, struct, switch, this, typedef, union, virtual, void, volatile, while

Deklaracja, inicjacja, definiowanie zmiennych.

DEKLARACJA

np. int a oznacza mniej więcej, jakbyśmy powiedzieli kompilatorowi: —Jakbyś gdzieś spotkał nazwę a to wiedz, że jest to zmienna typu intl. Określa typ wartości, jakie może przechowywać dana stała lub zmienna.

INICJACJA

to pierwsze przypisanie stałej lub zmiennej. Powoduje przydzielenie pamięci.

DEFINICJI

to jednocześnie deklaracja i inicjacja.

Język C/C++ operuje sześcioma podstawowymi typami danych:

- **char** (znak, numer znaku w kodzie ASCII) - 1 bajt;
- **int** (liczba całkowita) - 2 bajty;
- **float** (liczba z pływającym przecinkiem) - 4 bajty;
- **double** (podwójna ilość cyfr znaczących) - 8 bajtów;
- **bool** (wartość logiczna, prawda lub fałsz)
- **void** (nieokreślona) 0 bajtów.

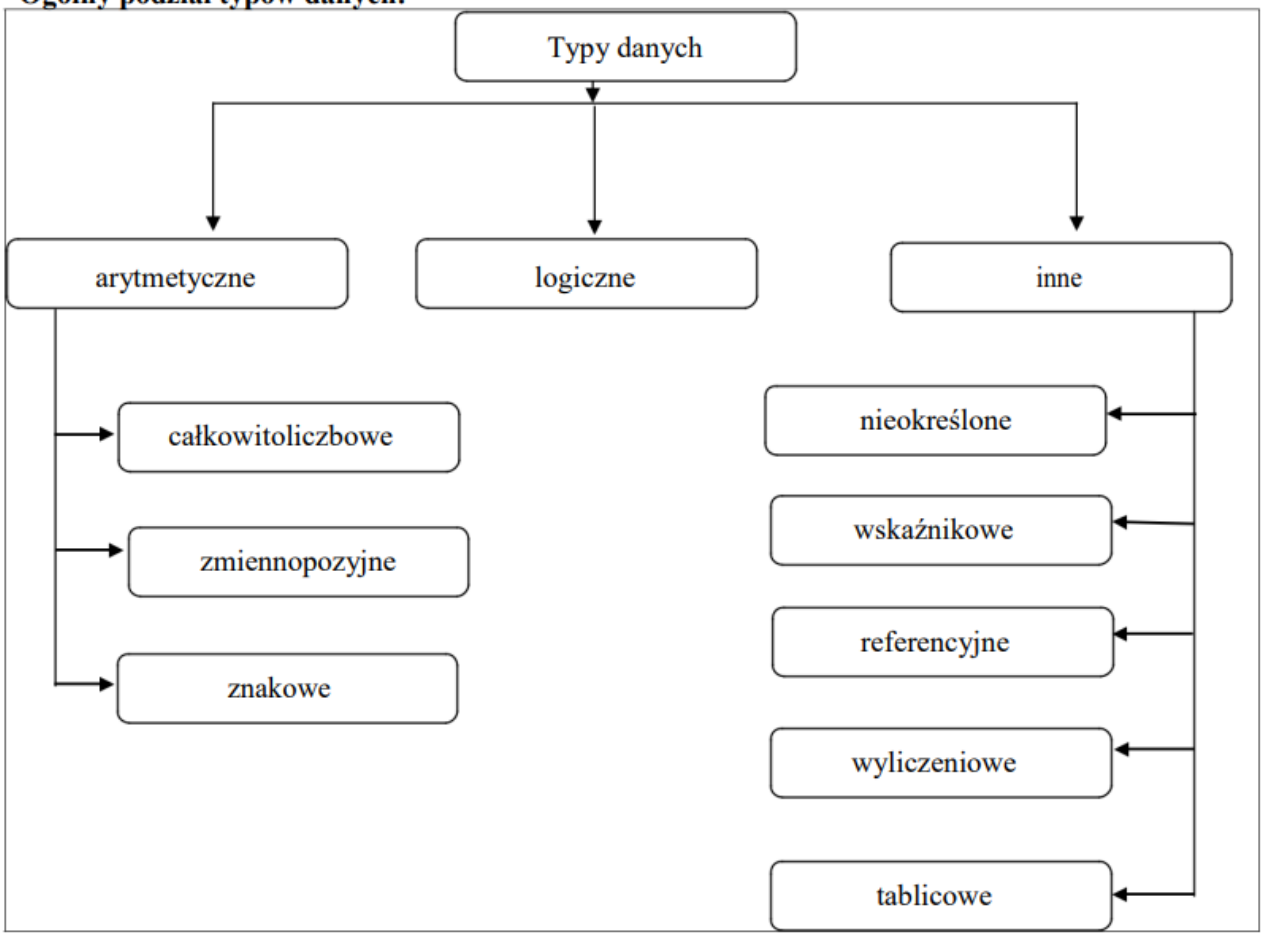
Podstawowe typy danych mogą być stosowane z jednym z czterech modyfikatorów:

- **signed** - ze znakiem
- **unsigned** - bez znaku
- **long** - długi
- **short** - krótki

Zakres wartości przedstawiono w Tabeli poniżej.

	Typ	Typowy rozmiar w bajtach	rozmiar
1	char signed	1 bajty	-128 do 127
2	unsigned char	1 bajty	0 do 255
3	unsigned short	2 bajty	0 do 65535
4	Int	2 bajty	-32768 do 32767
5	unsigned int	4 bajty	0 do 65535
6	long int	4 bajty	-2147483648 do 2147483647
7	signed short	2 bajty	od -32768 do 32767
8	signed int	4 bajty	Od -2147483648 do 2147483647
9	unsigned long int	4 bajty	0 do 4294967295
10	signed long int	4 bajty	-2147483648 do 2147483647
11	float	4 bajty	3.4 E- 38 do 3.4e+38
12	double	8 bajtów	1.7 E-308 do 1.7 E+308
13	long double	10 bajtów	1.2 E-4932 do 1.2E+4932
14	bool	1	true, false
15	unsigned short int	2 bajty	0 do 65535
16	unsigned long int	4 bajty	Od 0 do 4294967295
18	unsigned long long int	8 bajtów	Od 0 do 18446744073709551615
19	unsigned long long	8 bajtów	Od 0 do 18446744073709551615
20	short	2 bajty	Od -32768 do 32767
21	short int	2 bajty	Od -32768 do 32767
22	signed	4 bajty	Od -2147483648 do 2147483647
23	long	4 bajty	-2147483648 do 2147483647
24	signed long	4 bajty	-2147483648 do 2147483647
25	long long	8 bajtów	Od -9223372036854775808 do 9223372036854775807
26	long long int	8 bajtów	Od -9223372036854775808 do 9223372036854775807
27	signed long long	8 bajtów	Od -9223372036854775808 do 9223372036854775807
28	signed long long int	8 bajtów	Od -9223372036854775808 do 9223372036854775807

Ogólny podział typów danych:



Stałe

Stała to taka zmienna, której wartość można przypisać tylko raz.

const float PI = 3.14159;

nie można przypisać w programie żadnej innej wartości, innymi słowy zapis: jest jednocześnie DEKLARACJĄ, DEFINICJĄ i ZAINICJOWANIEM stałej PI.

Przykład :

float a,b,c;	(DEKLARACJA)
const float euler = 2.7	(DEFINICJA)
y = 441;	(ZAINICJOWANIE zmiennej)

Sposoby zapisu liczb zmiennoprzecinkowych

Sposób 1
zapis części ułamkowej pisanej po kropce np. 13.345
Sposób 2
sposób wykładniczy 34.6e4=34.6*10^4=34.6*10000=346000

Przykład 5

Temat:

Przykład demonstruje definiowanie zmiennych, wczytywanie ich wartości z klawiatury oraz operacje na nich. Całość z użyciem strumieni.

```
/* Oto program pokazujący operacje wczytywania z klawiatury i obliczania wartości wyrażenia */

#include <iostream>
#include <conio.h>
#include <cstdlib>

using namespace std;

int main(int argc, char *argv[])
{
   float metry;
   float kilometry;
   float k=1000;

   cout<<"Podaj liczbe
   metrow:"; cin>>metry;                    //wczytanie z klawiatury

   kilometry=metry/k;

   cout<<"Oto wynik przekształcenia:\n"<<metry<<" metrow to "<<kilometry<<" kilometrow.";
   cout<<endl;

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

Wytłumaczenie instrukcji użytych w przykładzie powyżej.

Definicje zmiennych

Pierwsza zmienna:

Instrukcja **float metry ;** powoduje, że tworzymy zmienną o nazwie **metry**, która jest typu **float**. Typ ten służy do przechowywania liczb zmiennoprzecinkowych czyli takich, które mają przecinki.

Druga zmienna:

Tworzymy zmienną **kilometry**, która jest typu **float**.

Trzecia zmienna:

Definiujemy w niej zmienną k typu float i **od razu wpisujemy do niej wartość 1000**. Uwaga: W momencie utworzenia zmiennych tzw. lokalnych znajduje się w nich wartość przypadkowa.
W C++ można definiować zmienne wtedy, kiedy się zorientujemy, że są nam potrzebne.

Opis instrukcji

cin >> coś tam ; oznacza wczytanie z klawiatury do zmiennej **coś tam**. Jak widać, jest to ruch tak jakby odwrotny do wypisywania – tak nam sugerują strzałki.

Przeliczenia na kilometry. Dzielenie to /.

Po przeliczeniu wypisujemy na ekran wynik.
_\n‘ – nowa linia

Instrukcja **cin** → wczytywanie (pobierania) danych do programu z klawiatury lub pliku zewnętrznego.

np. cin>>promien;

Wczytanie z klawiatury liczby do zmiennej o nazwie **promien**.

np. cin>>x>>y

Wczytanie z użyciem jednej instrukcji cin dwóch zmiennych o nazwach **x** i **y**.

Operacja przypisania
do przypisania używa się znaku = a nie := jak w pascalu.
np.
p = (a+b)/2*h;
x = y =10; przypisanie zmiennym x i y wartości 10

Inkrementacja
Pojęcie to oznacza zwiększanie o jeden wartości zmiennej
np.
x++
wartość zmiennej x została zwiększona o jeden

Dekrementacja
Pojęcie to oznacza zmniejszanie o jeden wartości zmiennej
np.
y--
wartość zmiennej y została zmniejszona o jeden

OPERATORY ARYTMETYCZNE języka C++.

Operator	Nazwa	Tłumaczenie	Działanie
+	ADDition	Dodawanie	Suma liczb
-	SUBstraction	Odejmowanie	Różnica liczb
*	MULtiplication	Mnożenie	Iloczyn liczb
/	DIVision	Dzielenie	Iloraz liczb
%	Moduluj	Dziel Modulo	Reszta z dzielenia

Operatory relacji języka C++.

Operator	Nazwa	przykład	wynik
==	Równe	34==(37-3)	prawda
!=	różne od	9!=(15-6)	Fałsz
<	Mniejsze	10<11	prawda
>	Większe	9>8	fałsz
<=	mniejsze lub równe	7<=7	prawda
>=	większe lub równe	3>=4	fałsz

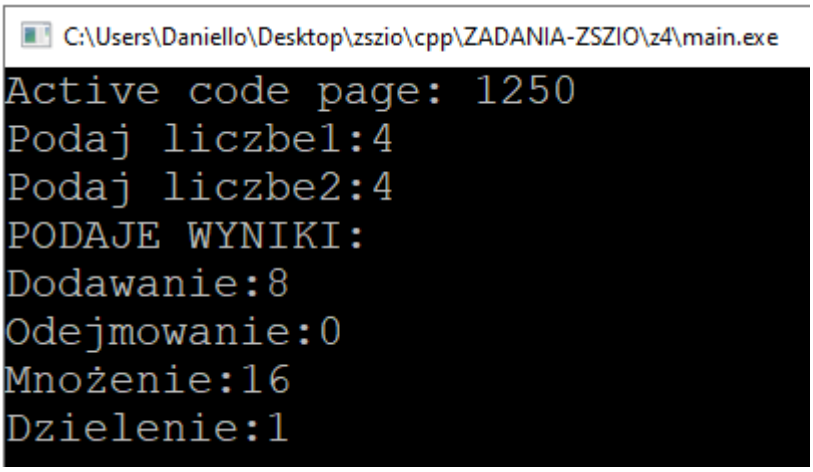
Operatory logiczne języka C++.

AND → && koniunkcja **i**
OR → || alternatywa **lub**
NOT → ! zaprzeczenie **nie**

np.
if (a>0 && a<100) printf("Dwucyfrowa"); else printf("100+");

Zadanie 4

Wykonaj program „prosty kalkulator” którego zadaniem będzie wczytanie 2 liczb następnie przedstawienie dodawanie, odejmowania, mnożenia, dzielenia (do wykonania zadania skorzystaj z przykładu 5)

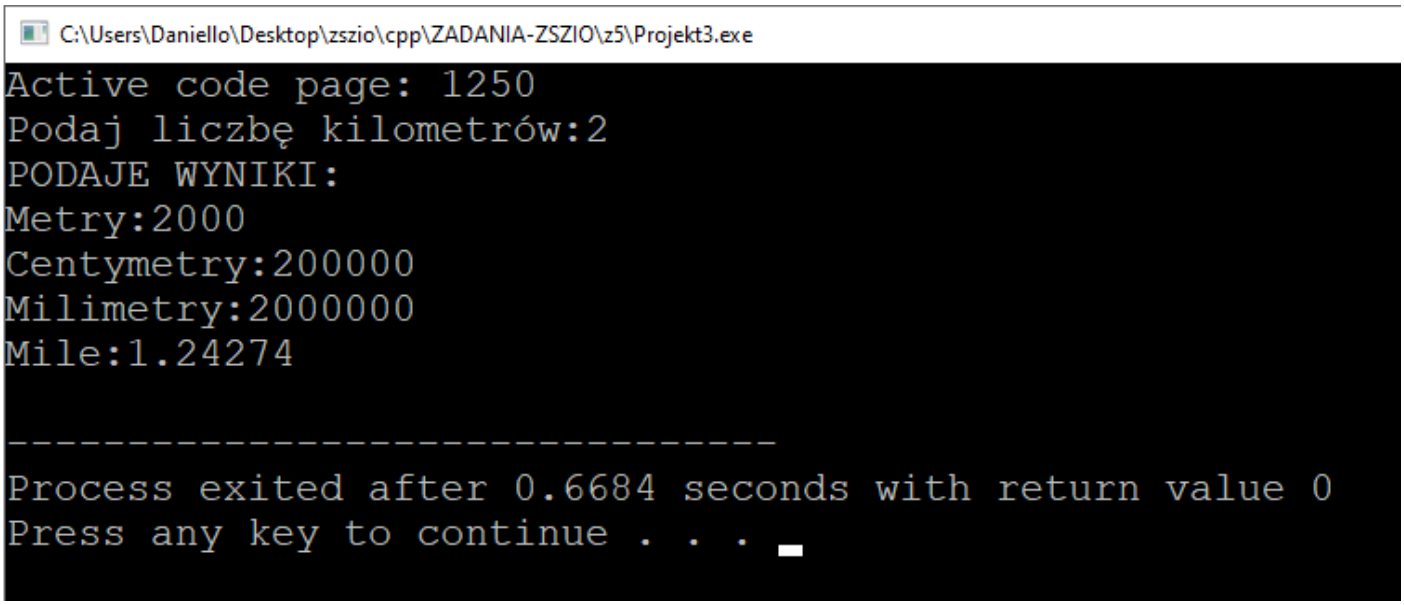


Zadanie 5

Wykonaj program którego zadaniem będzie zamiana kilometrów na metry, centymetry, milimetry, mile.

(Aby wykonać prawidłowo zadanie zwróć uwagę na typy zmiennych)

(1 mila 0,621371192)



Zadanie 6

Wykonaj program „Kasa fiskalna” którego zadaniem będzie przyjęcie danych klienta, zamówienia, obliczenie sumy.