



**POLITECHNIKA LUBELSKA
WYDZIAŁ ELEKTROTECHNIKI
I INFORMATYKI**

**KIERUNEK STUDIÓW
INFORMATYKA**

*MATERIAŁY DO ZAJĘĆ
LABORATORYJNYCH*

PROGRAMOWANIE STRUKTURALNE

Autor:
dr inż. Elżbieta Miłośz

Lublin 2019



**Fundusze
Europejskie**
Wiedza Edukacja Rozwój



**Rzeczpospolita
Polska**

Unia Europejska
Europejski Fundusz Społeczny



INFORMACJA O PRZEDMIOCIE

Cele przedmiotu:

- Poznanie podstaw programowania strukturalnego na przykładzie języka C.
- Praktyczna nauka posługiwania się specyficznymi mechanizmami programowania w języku C.
- Poznanie metod korzystania z biblioteki standardowej.

Efekty kształcenia w zakresie umiejętności:

- Potrafi posługiwać się dokumentacją opisującą bibliotekę języka C, wyszukiwać niezbędne informacje w literaturze, także w języku angielskim.
- Potrafi opisać w sposób niesformalizowany wymagania wobec aplikacji o charakterze strukturalnym.
- Potrafi zaprojektować aplikację strukturalną o średnim i dużym stopniu złożoności.
- Potrafi wybrać i zastosować w praktyce właściwy sposób organizacji prac programistycznych, w tym technikę testowania aplikacji.

Literatura do zajęć:

Literatura podstawowa

1. Montusiewicz J., Miłosz E., Jarosińska-Caban M., Podstawy programowania w języku C. Ćwiczenia laboratoryjne, Wydawca PL, Lublin 2015.
2. Stabrowski M. M., Język C w przykładach, Wydawnictwo WSEI, Lublin 2011.
3. Schildt H., Programowanie: C, Wydawnictwo RM, Warszawa 2002.
4. Kernighan B. W., Ritchie D., Język C, WNT, 1987.

Literatura uzupełniająca

1. King. K. N., Język C. Nowoczesne programowanie, Helion, Gliwice 2011.
2. Prata S., Szkoła programowania. Język C, Wydawnictwo Robomatic, Wrocław, 1999.

Metody i kryteria oceny:

- Oceny częściowe:
 - Przygotowanie merytoryczne do zajęć laboratoryjnych na podstawie: wykładów, literatury, pytań kontrolnych do zajęć.
 - Frekwencja i kreatywność na zajęciach: próg zaliczeniowy 80%.
 - Dwa kolokwia: próg zaliczeniowy 60%.
- Ocena końcowa - zaliczenie przedmiotu:
 - Pozytywne oceny częściowe.
 - Ewentualne dodatkowe wymagania prowadzącego zajęcia.



Plan zajęć laboratoryjnych:

Lab1.	Wprowadzenie do programowania strukturalnego w języku C. Wprowadzenie do środowiska programistycznego.
Lab2.	Podstawy tworzenia algorytmów, schematy Nassi-Schneidermana.
Lab3.	Proste programy imperatywne. Wprowadzanie i wyprowadzanie danych. Zmienne różnych typów. Instrukcja przypisania. Wyrażenia.
Lab4.	Proste programy strukturalne. Funkcje standardowe i funkcje własne. Argumenty funkcji.
Lab5.	Instrukcje warunkowe IF, IF...ELSE. Operator warunkowy. Instrukcja wyboru SWITCH. Instrukcja BREAK.
Lab6.	Instrukcje iteracyjne WHILE, DO...WHILE. Instrukcja CONTINUE.
Lab7.	Instrukcja iteracyjna FOR.
Lab8.	Kolokwium 1.
Lab9.	Funkcje z argumentami wskaźnikowymi.
Lab10.	Złożone typy danych. Tablice statyczne jedno i wielowymiarowe.
Lab11.	Dynamiczna alokacja pamięci. Tablice dynamiczne.
Lab12.	Łańcuchy znakowe i funkcje łańcuchowe.
Lab13.	Złożone typy danych. Struktury i unie.
Lab14.	Złożone typy danych. Pliki.
Lab15.	Kolokwium 2.

Zajęcia laboratoryjne zawierają pytania kontrolne, które student powinien opanować przed przystąpieniem do zajęć, zestawy 1 lub 2 zadań do analizy (zawierające rozwiązanie w postaci algorytmu NS lub kodu źródłowego w języku C) oraz zestawy 10 zadań (5 podstawowych + 5 dodatkowych) do wykonania (zawierające treść zadania i polecenia do wykonania).

Przygotowanie do zajęć laboratoryjnych może być realizowane przez studenta na podstawie wykładów i zalecanej literatury. Dodatkowo, w każdym laboratorium umieszczono krótkie kompendium wiedzy związane z zakresem realizowanego laboratorium.

Prowadzący zajęcia ocenia przygotowanie merytoryczne studentów do zajęć laboratoryjnych, wybiera zadania do realizacji na laboratorium i ocenia frekwencję i kreatywność studentów na zajęciach.

Zestawy zadań zostały opracowane na podstawie podręcznika: *Montusiewicz J., Miłosz E., Jarońska-Caban M., Podstawy programowania w języku C. Ćwiczenia laboratoryjne*, Wydawca PL, Lublin 2015.

LABORATORIUM 1. WPROWADZENIE DO PROGRAMOWANIA STRUKTURALNEGO W JĘZYKU C. WPROWADZENIE DO ŚRODOWISKA PROGRAMISTYCZNEGO.

Cel laboratorium:

Zaznajomienie ze strukturą prostego programu w języku C. Zaznajomienie z procesem pisania i uruchamiania programów w zintegrowanych środowiskach programistycznych Dev-C++ i Code::Blocks. Nabycie praktycznych umiejętności pracy w środowiskach: Dev-C++ i Code::Blocks.

Zakres tematyczny zajęć:

- struktura programu w języku C,
- elementy środowiska programistycznego (Dev-C++ lub Code::Blocks),
- sposób tworzenia i zapisu nowego projektu,
- tworzenie kodu źródłowego programu, kompilacja i wykonanie,
- procesy edycji kodu źródłowego, kompilacji, linkowania, uruchomienia programu,
- błędy kompilacji,
- plik źródłowy a plik wykonywalny.

Kompendium wiedzy:

Program w języku C – zasady:

- Program w języku C jest zbiorem modułów.
- Każdy moduł składa się z globalnych deklaracji typów, zmiennych, funkcji.
- Dokładnie jeden moduł zawiera definicję funkcji głównej (main).
- Wykonanie programu polega na opracowaniu jego globalnych deklaracji, a potem na wykonaniu instrukcji funkcji main.
- Zakończenie programu następuje po wykonaniu instrukcji powrotu (return) w funkcji main lub po powrocie z funkcji exit.
- Instrukcje przeznaczone dla kompilatora (dyrektywy preprocesora) rozpoczynają się # i nie kończą średnikiem.
- Wielkość liter w identyfikatorach (nazwach własnych) ma znaczenie.
- Komentarze:

*/*komentarz w C */*

lub //komentarz w C i C++

Struktura programu w C

```
//dyrektywy preprocesora
int main() //naglowek funkcji głównej
{
//deklaracje
//instrukcje
return 0; //instrukcja powrotu
}
```



Proces programowania:

1. Określenie celów programu.
2. Zaprojektowanie algorytmu programu.
3. Utworzenie projektu i edycja kodu źródłowego programu.

Projekt w środowiskach programistycznych - zbiór modułów kodu źródłowego i innych plików, które po kompilacji i łączeniu stają się pojedynczym plikiem EXE, czyli programem (zalecany oddzielny katalog).

4. Kompilacja i łączenie.

Kompilacja – tłumaczenie kodu źródłowego (*.c) na kod maszynowy (*.obj).

Łączenie – konsolidacja skompilowanych plików z bibliotekami do pliku wykonywalnego (*.exe).

5. Uruchomienie programu.
6. Testowanie i usuwanie błędów.
7. Pielęgnowanie i modyfikacja programu.

Pytania kontrolne:

1. Objaśnij strukturę prostego programu w języku C.
2. Jak utworzyć nowy program w środowisku programistycznym Dev-C++ (Code::Blocks)?
3. Jak uruchomić program?
4. Na czym polega proces kompilacji?
5. Na czym polega proces łączenia?

Zadania do analizy

Zadanie 1.1. Struktura prostego programu w języku C

Przeanalizuj kod źródłowy prostego programu w języku C. Zwróć uwagę na komentarze.

```
#include <stdio.h>           /*dyrektywy kompilatora*/
#include <stdlib.h>          /*dyrektywy kompilatora*/

int main(int argc, char *argv[]) {    /*funkcja glowna*/
    int ocena;                      /*definicja zmiennej o nazwie ocena*/
    ocena=5;                        /*przypisanie zmiennej wartości 5 */
    /*wyswietlenie na ekranie*/
    printf("Zaczynam programowac w jezyku C\n");
    printf("Z przedmiotu \"Programowanie strukturalne\"
           otrzymam ocene %d\n",ocena);
    return 0;
}
```

Zadanie 1.2. Zintegrowane środowiska programistyczne Dev-C++ i Code::Blocks

Przeanalizuj sposób pracy w zintegrowanym środowisku programistycznym **Dev-C++**:

- Pakiet instalacyjny Dev-C++: <http://www.bloodshed.net/devcpp.html>.
- Tworzenie nowego projektu: *Plik/ Nowy projekt/ Console Application* z opcją *Projekt C*.
- Zapis plików projektu: *Plik/Zapisz (Zapisz jako/Zapisz projekt jako /Zapisz wszystko)*.
- Otwarcie projektu/pliku: *Plik/Otwórz projekt lub plik*.
- **Uruchomienie programu:**
 - *Uruchom/Kompiluj* - kompilacja kodu aktualnego pliku źródłowego.
 - *Uruchom/ Uruchom* - uruchomienie programu.
 - *Uruchom/ Kompiluj i uruchom* - kompilacja przyrostowa plików źródłowych i uruchomienie.
 - *Uruchom/ Przebuduj wszystko* - skompilowanie od zera wszystkich plików źródłowych.

Przeanalizuj sposób pracy w zintegrowanym środowisku programistycznym **Code::Blocks**:

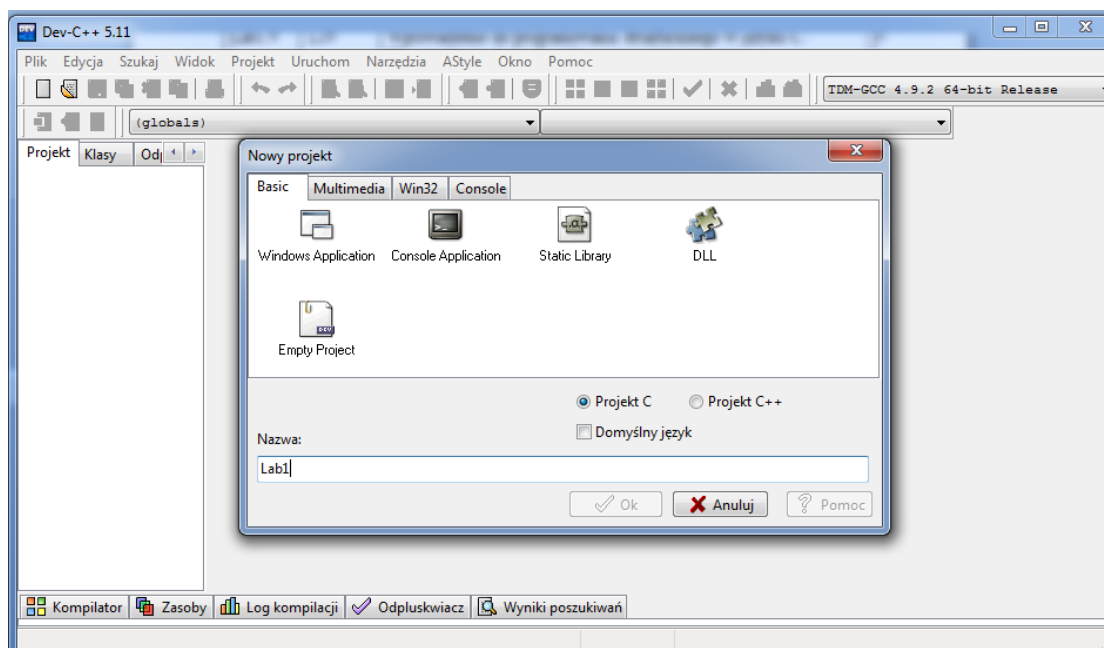
- Pakiet instalacyjny Code::Blocks: www.codeblocks.org.
- Tworzenie nowego projektu: *File/New/Project/Console Application* z opcją *C*.
- Zapis plików projektu: *File/Save file (Save file as /Save Project as /Save all files)*.
- Otwarcie projektu/pliku: *File/Open*.
- **Uruchomienie programu:**
 - *Build/Build* - kompilacja kodu aktualnego pliku źródłowego.
 - *Build/Run* - uruchomienie programu.
 - *Build/Build and run* - kompilacja przyrostowa plików źródłowych i uruchomienie.
 - *Build/Rebuild* - skompilowanie od zera wszystkich plików źródłowych.

Zadania do wykonania

Zadanie 1.3. Uruchomienie prostego programu w zintegrowanym środowisku programistycznym Dev-C++ i testowanie błędów

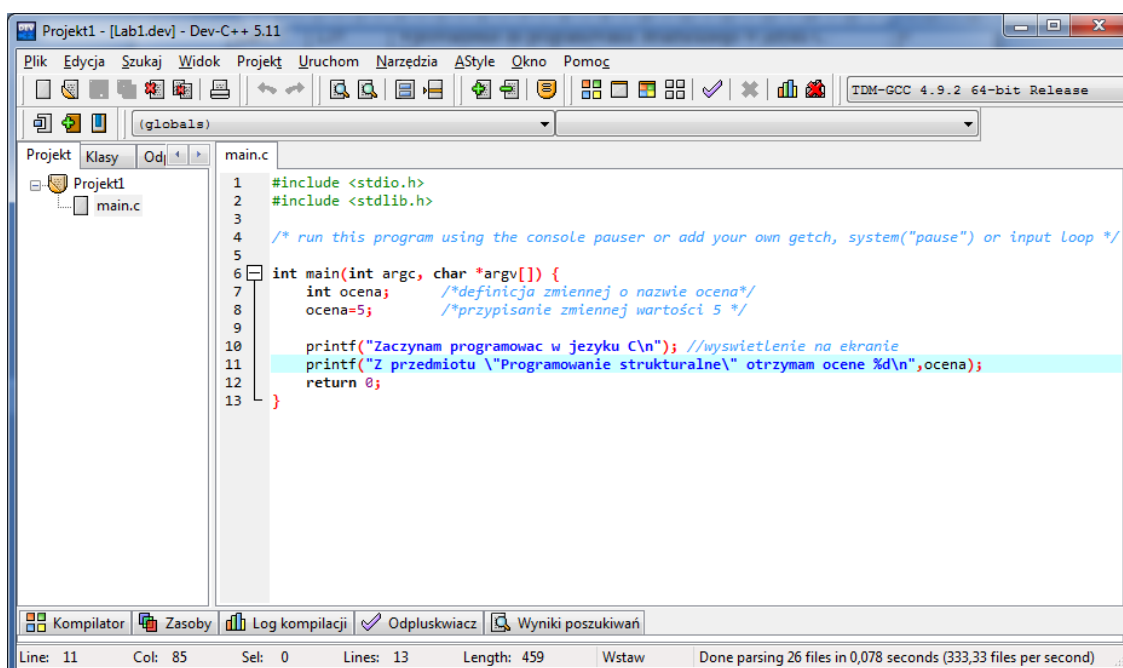
- Uruchom środowisko programistyczne Dev-C++.
- Utwórz nowy projekt w C jako aplikację konsolową (Rys.1.1).





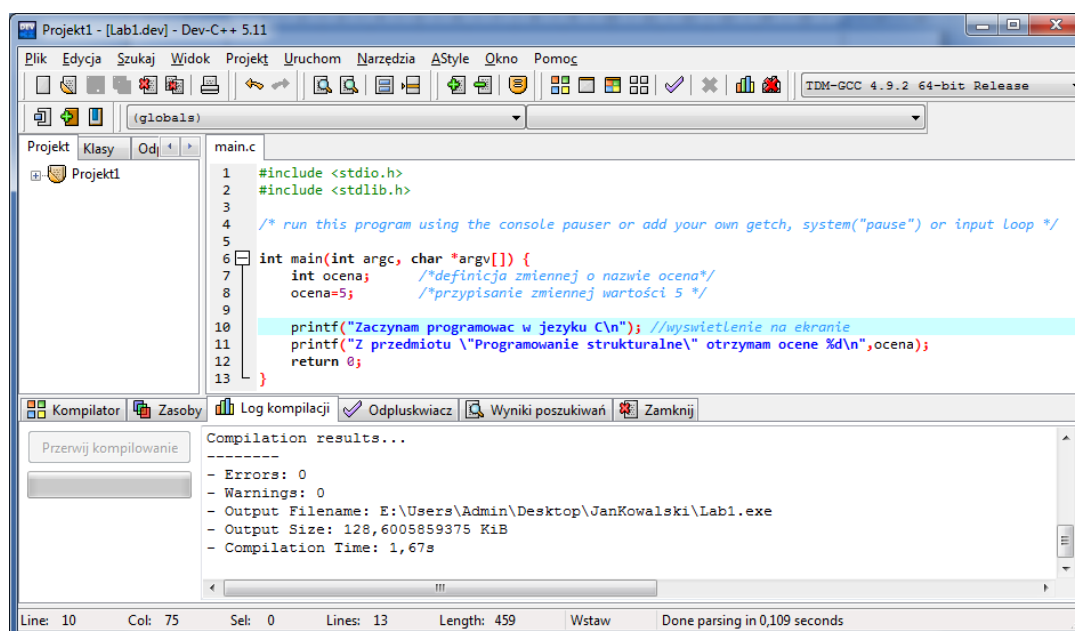
Rys. 1. 1 Tworzenie projektu aplikacji konsolowej w Dev-C++

- Zapisz plik pod nazwą Lab1 w katalogu o swoim nazwisku na pulpicie.
- Zapoznaj się z paskami narzędzi i menu środowiska programistycznego Dev-C++.
- Do edytora kodu w pliku main.c wprowadź kod źródłowy programu z zadania 1.1 (Rys. 1.2).



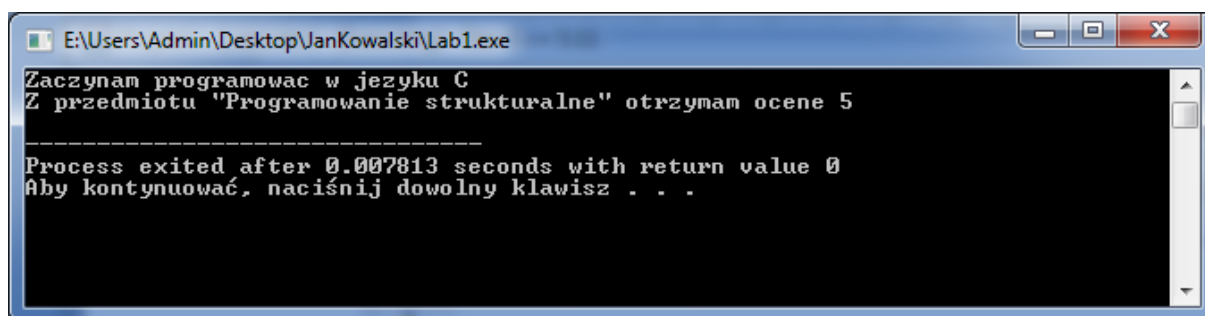
Rys. 1. 2 Kod źródłowy programu z zadania 1.1 w Dev-C++

- Wykonaj kompilację programu poleceniem *Uruchom/Kompiluj* i sprawdź jej poprawność (Rys. 1.3).



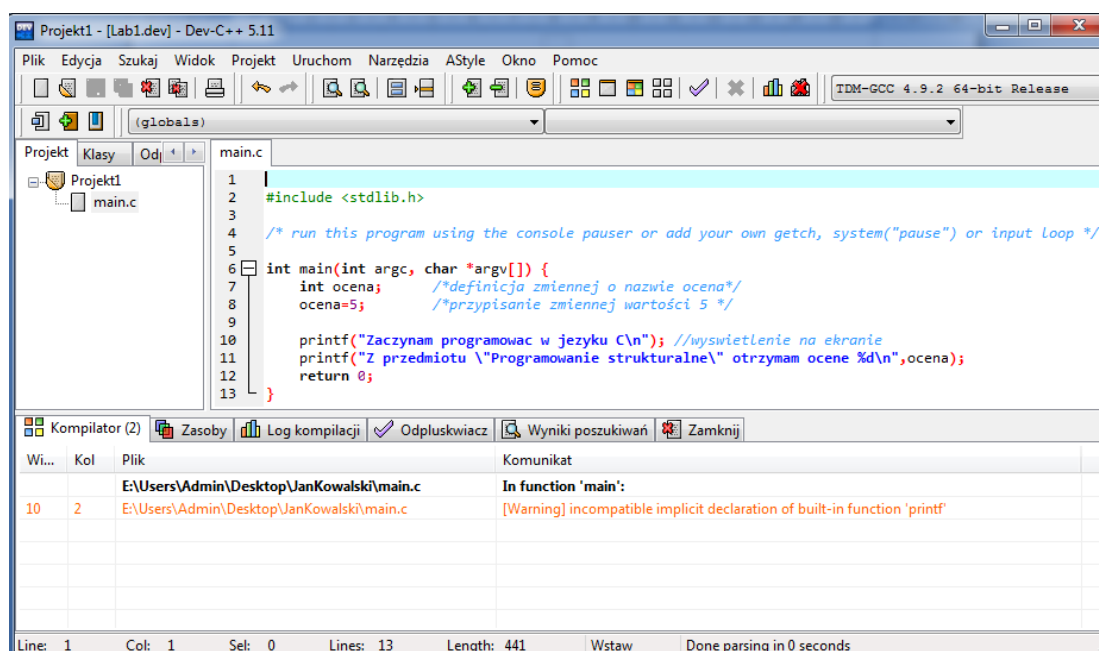
Rys. 1. 3 Informacja o wyniku kompilacji programu

- Uruchom program (*Uruchom/Uruchom*) i przeanalizuj jego rezultat (Rys. 1.4).



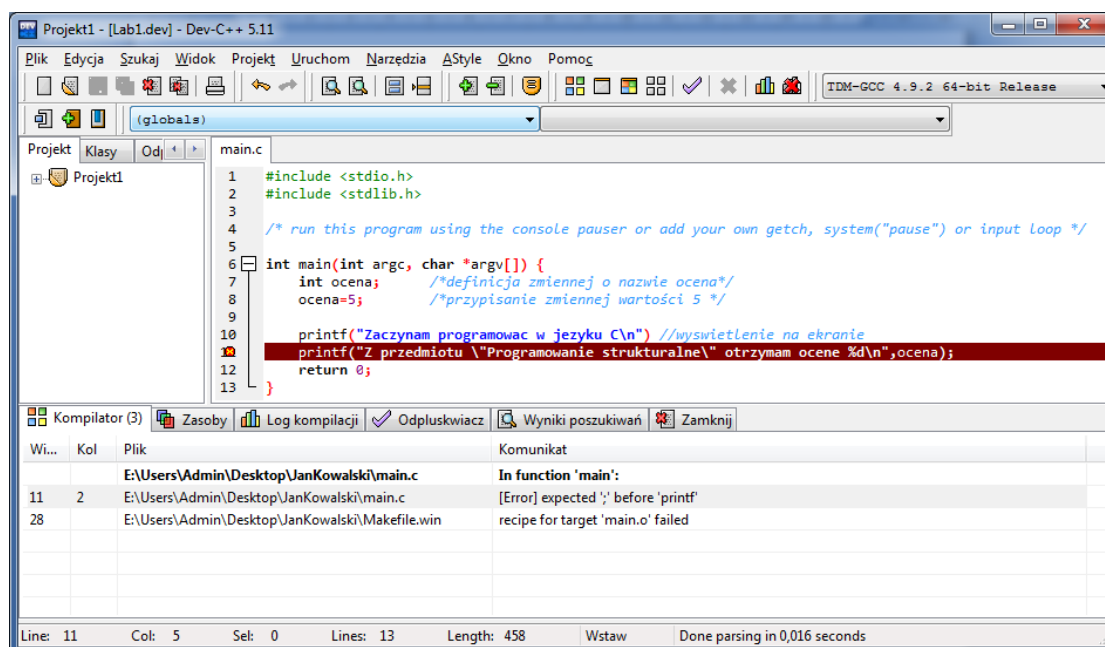
Rys. 1. 4 Wynik uruchomienia programu

- Wprowadź błąd w kodzie np. usuń wiersz 1 (brak deklaracji pliku nagłówkowego `stdio.h`) i ponownie skompiluj program. Przeanalizuj informację o rodzaju błędu (Rys.1.5).



Rys. 1. 5 Błąd łączenia

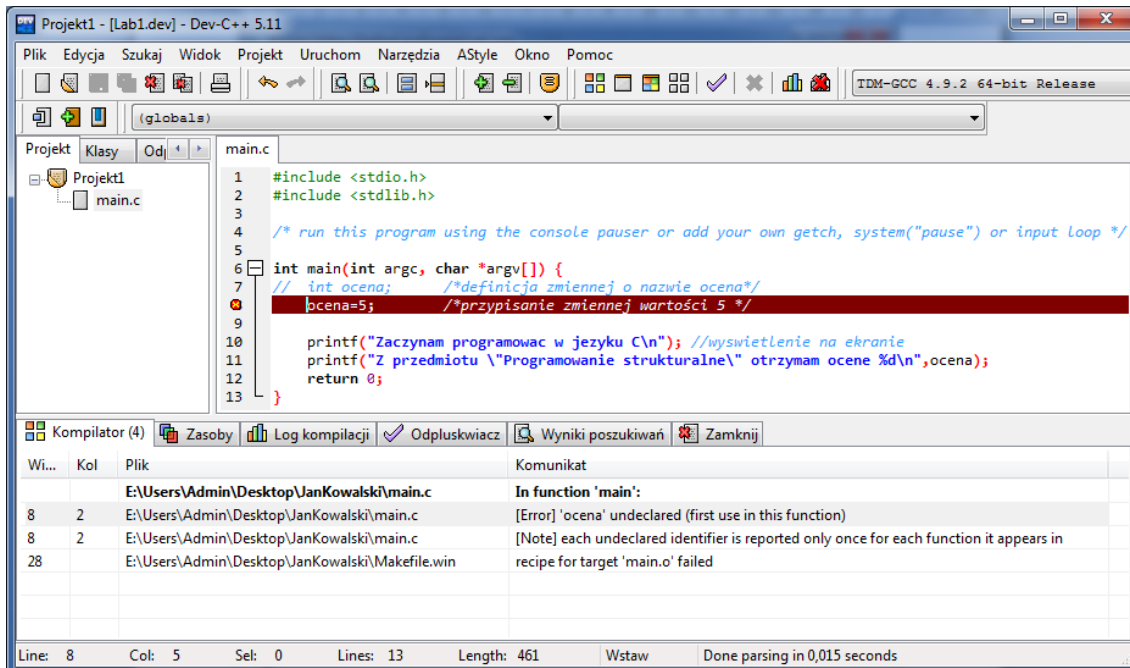
- Popraw błąd w kodzie źródłowym i ponownie uruchom program poleceniem *Uruchom/ Kompiluj i uruchom*.
- Wprowadź kolejny błąd w kodzie np. usuń średnik na końcu polecenia w wierszu 10 i ponownie skompiluj program. Przeanalizuj informację kompilatora o rodzaju błędzie (Rys.1.6).



Rys. 1. 6. Błąd składni

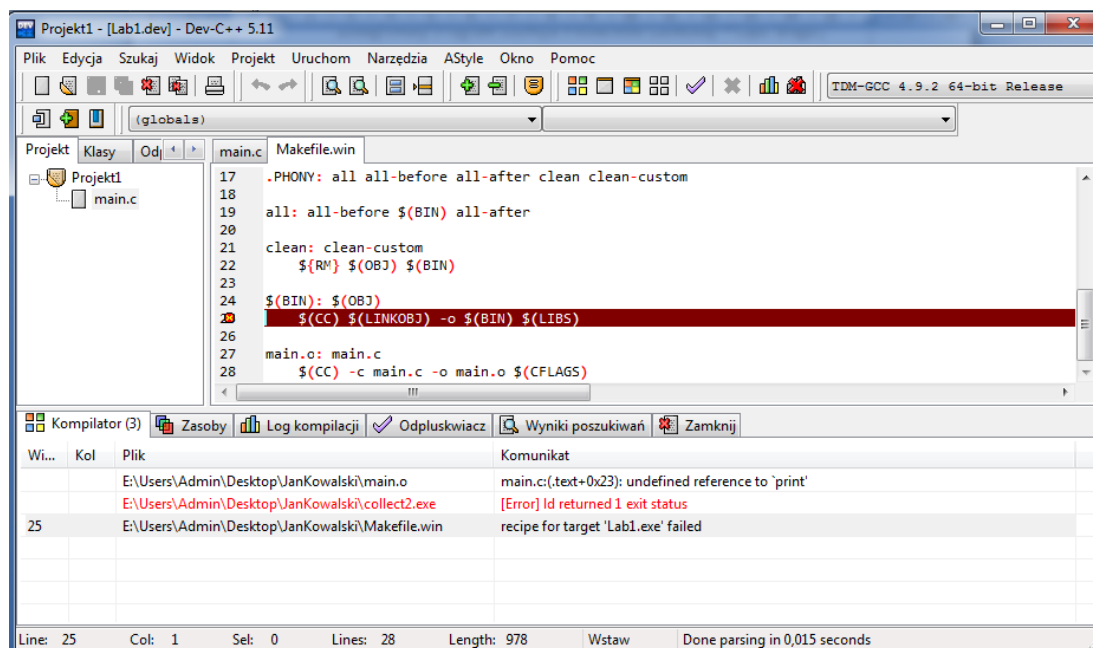
- Popraw błąd w kodzie źródłowym i ponownie uruchom program poleceniem *Uruchom/ Kompiluj i uruchom*.

- Wprowadź kolejny błąd w kodzie np. zakomentuj wiersz 7 i ponownie skompiluj program. Przeanalizuj informację kompilatora o rodzaju błędu (Rys.1.7).



Rys. 1. 7 Błąd semantyczny

- Popraw błąd w kodzie źródłowym i ponownie uruchom program poleceniem *Uruchom/ Kompiluj i uruchom*.
- Znajdź odpowiednie ikony realizowanych poleceń menu *Uruchom* na pasku narzędziowym.
- Wprowadź kolejny błąd w kodzie np. w wierszu 10 usuń literę f z funkcji printf (zła nazwa funkcji) i ponownie skompiluj program. Przeanalizuj informację kompilatora o rodzaju błędu (Rys.1.8).

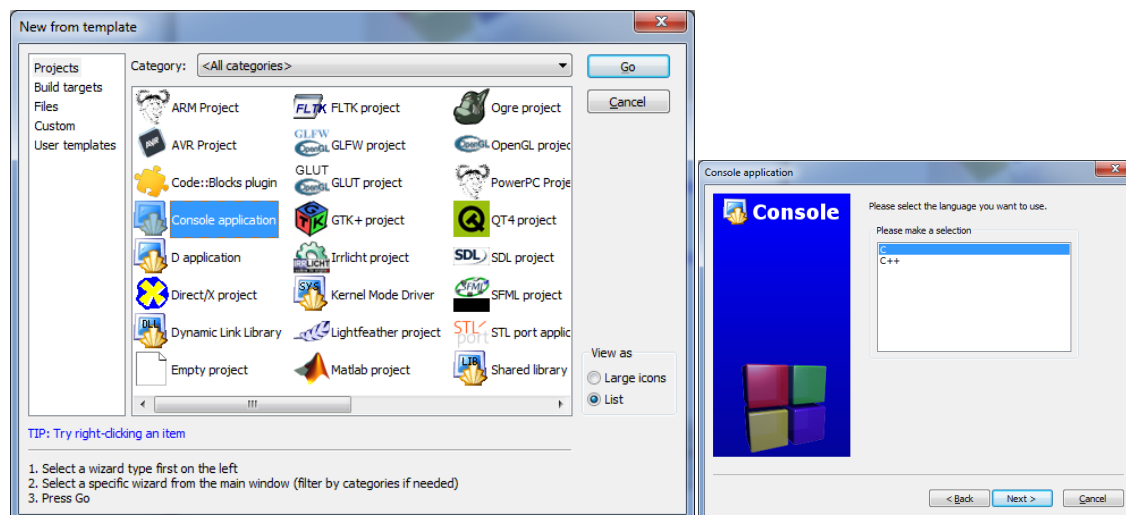


Rys. 1. 8 Błąd łączenia

- Popraw błąd w kodzie źródłowym i ponownie uruchom program przyciskiem .

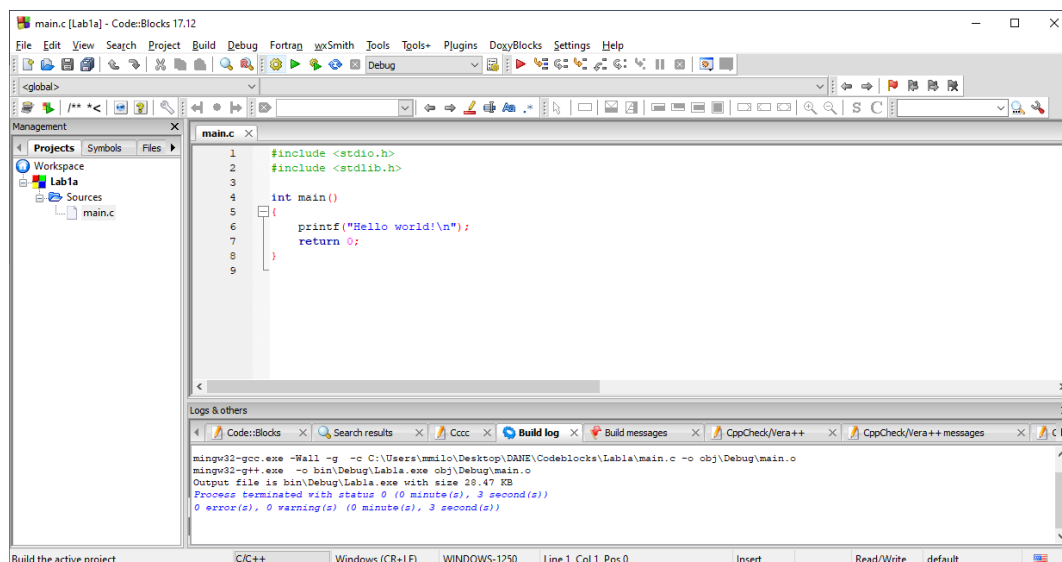
Zadanie 1.4. Uruchomienie prostego programu w zintegrowanym środowisku programistycznym Code::Blocks

- Uruchom środowisko programistyczne Code::Blocks.
- Utwórz nowy projekt w C jako aplikację konsolową (Rys.1.9).



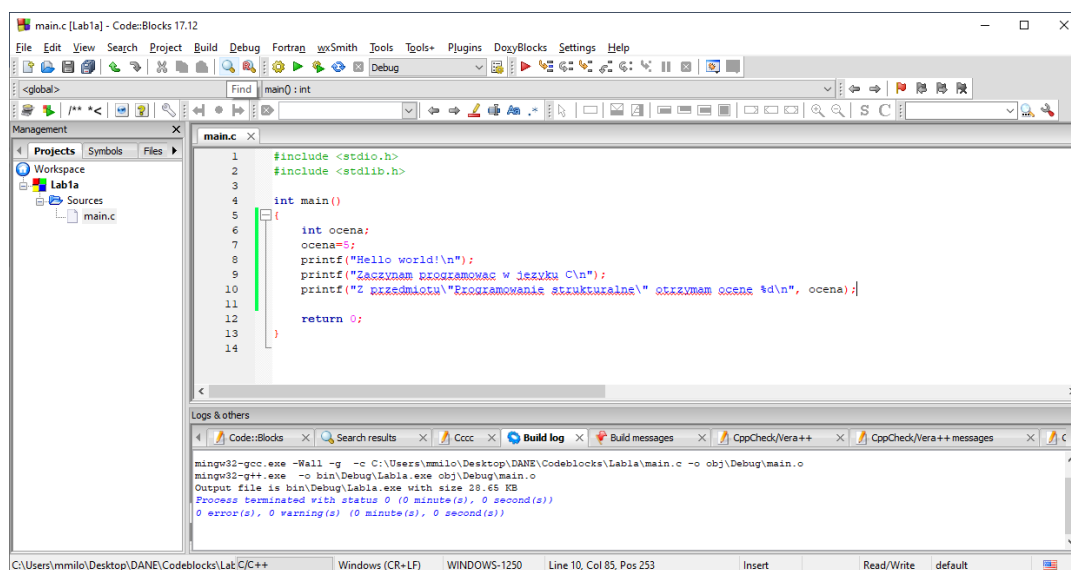
Rys. 1. 9 Tworzenie projektu aplikacji konsolowej w Code::Blocks

- Zapisz plik pod nazwą Lab1a w katalogu o swoim imieniu na pulpicie.
- Zapoznaj się z paskami narzędzi i menu środowiska programistycznego Code::Blocks (Rys.1.10).



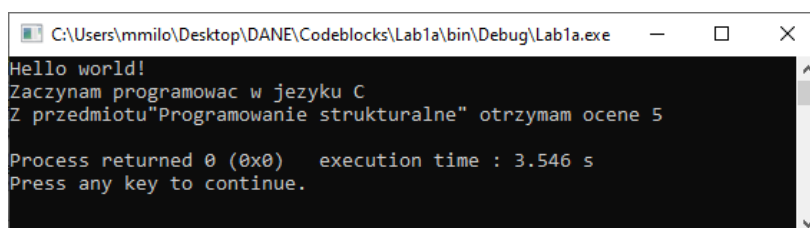
Rys. 1. 10 Nowy projekt w Code::Blocks

- Do edytora kodu w pliku `main.c` wprowadź kod źródłowy programu z zadania 1.1.
- Wykonaj kompilację programu poleceniem *Build/Build* i sprawdź jej poprawność (Rys. 1.11).



Rys. 1. 11 Kompilacja programu

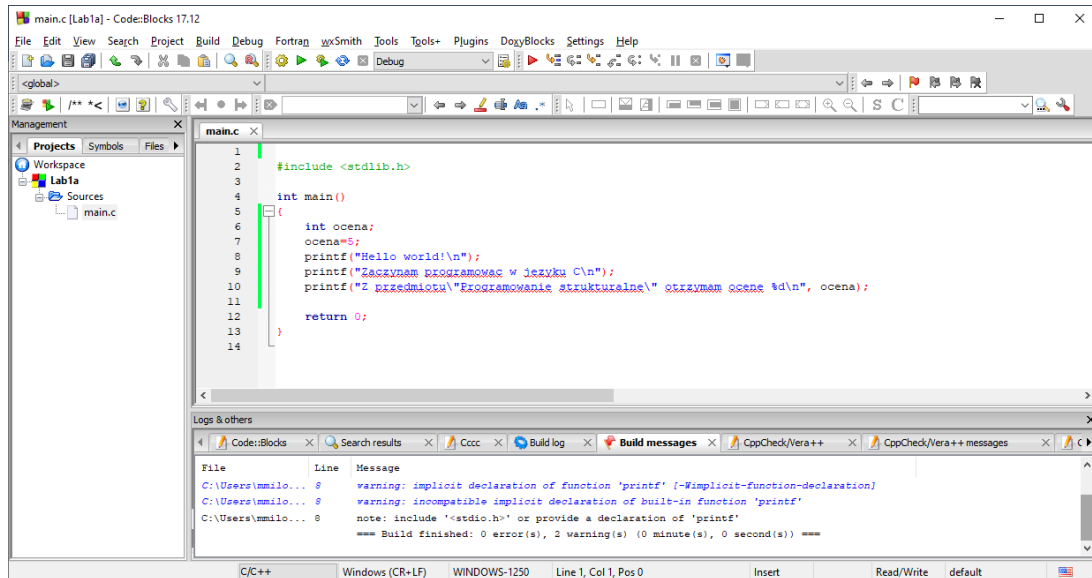
- Uruchom program i przeanalizuj jego rezultat (Rys.1.12).



Rys. 1. 12 Rezultat wykonania programu

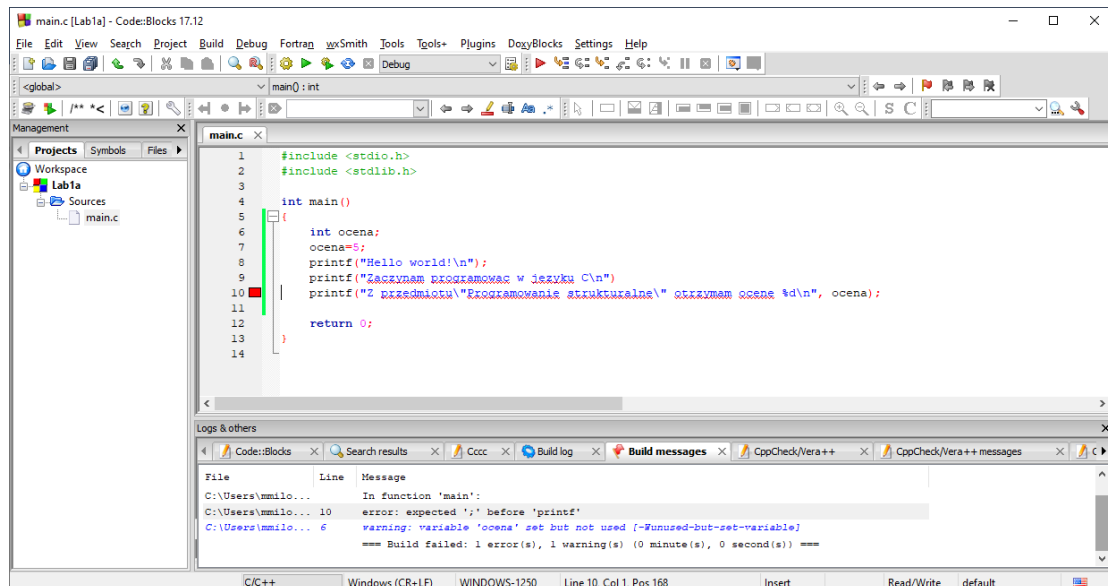


- Wprowadź celowo błąd w kodzie np. usuń wiersz 1 (brak deklaracji pliku nagłówkowego `stdio.h`) i ponownie skompiluj program. Przeanalizuj informację kompilatora o rodzaju błędu (Rys.1.13).



Rys. 1. 13. Błąd łączenia

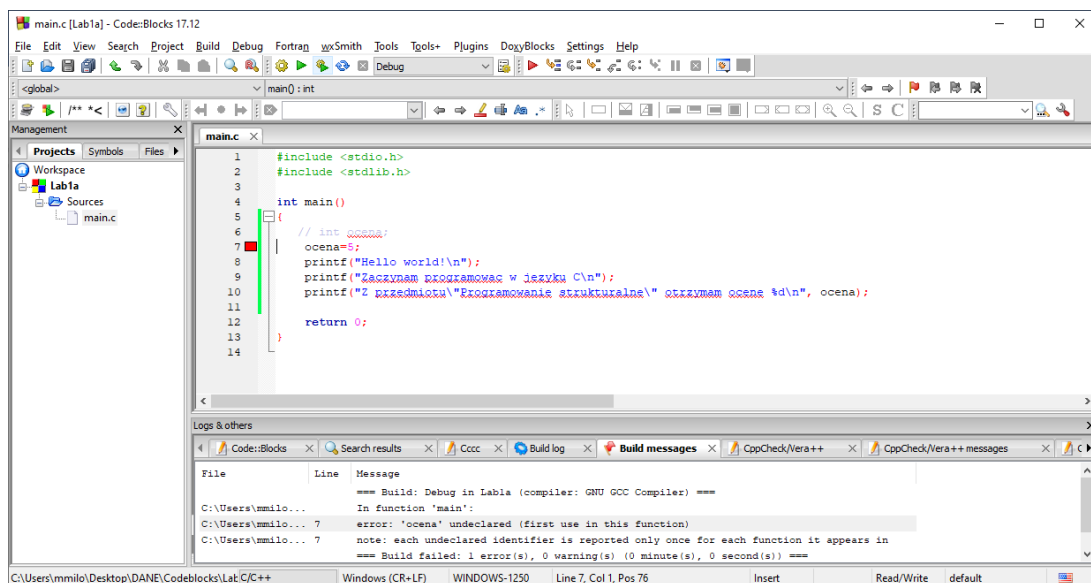
- Popraw błąd w kodzie źródłowym i ponownie uruchom program poleceniem *Build/Build and run*.
- Wprowadź kolejny błąd w kodzie: usuń średnik na końcu polecenia w wierszu 9 i ponownie skompiluj program. Przeanalizuj informację kompilatora o rodzaju błędu (Rys.1.14).




Rys. 1. 14 Błąd składni

- Popraw błąd w kodzie źródłowym i ponownie uruchom program poleceniem *Build/Build and run*.

- Wprowadź kolejny błąd w kodzie np. zakomentuj wiersz 6 i ponownie skompiluj program. Przeanalizuj informację kompilatora o rodzaju błędu (Rys.1.15).



Rys. 1. 15 Błąd semantyczny

- Znajdź odpowiednie ikony realizowanych poleceń menu *Build* na pasku narzędziowym.
- Popraw błąd w kodzie źródłowym i ponownie uruchom program przyciskiem .

Zadanie 1.4. Porównanie środowisk programistycznych Dev-C++ i Code::Blocks

- Na podstawie realizacji zadań 1.2 i 1.3 dokonaj porównania procesu edycji i uruchamiania programów w tych środowiskach.
- Zapoznaj się z opiniami programistów na temat wyboru środowiska programistycznego w języku C np. <http://cpp0x.pl/kursy/Kurs-C++/Poziom-1/Wybieramy-srodowisko-pracy/4>.



LABORATORIUM 2. PODSTAWY TWORZENIA ALGORYTMÓW, SCHEMATY NASSI-SCHNEIDERMANA.

Cel laboratorium:

Poznanie podstawowych sposobów zapisu algorytmu, w szczególności zasad tworzenia schematów zwartych (NS – Nassi_Schneidermana). Nabycie praktycznych umiejętności w tworzeniu i zapisie algorytmów.

Zakres tematyczny zajęć:

- pojęcie algorytmu,
- sposoby przedstawienia algorytmu,
- schematy NS (Nassi-Schneidermana).

Kompendium wiedzy:

Pojęcie algorytmu:

- a) klasyczne: *przepis na postępowanie rachunkowe*,
- b) rozszerzone: *opis obiektów* oraz *opis czynności*, które należy wykonać z tymi obiektami, aby osiągnąć *określony cel*; opis obiektów - deklaracje; opis czynności – instrukcje.

Sposoby przedstawienia algorytmu:

- a) słowny: lista kroków, pseudokod,
- b) graficzny: schematy blokowe, schematy zwarte Nassi-Schneidermana (NS).

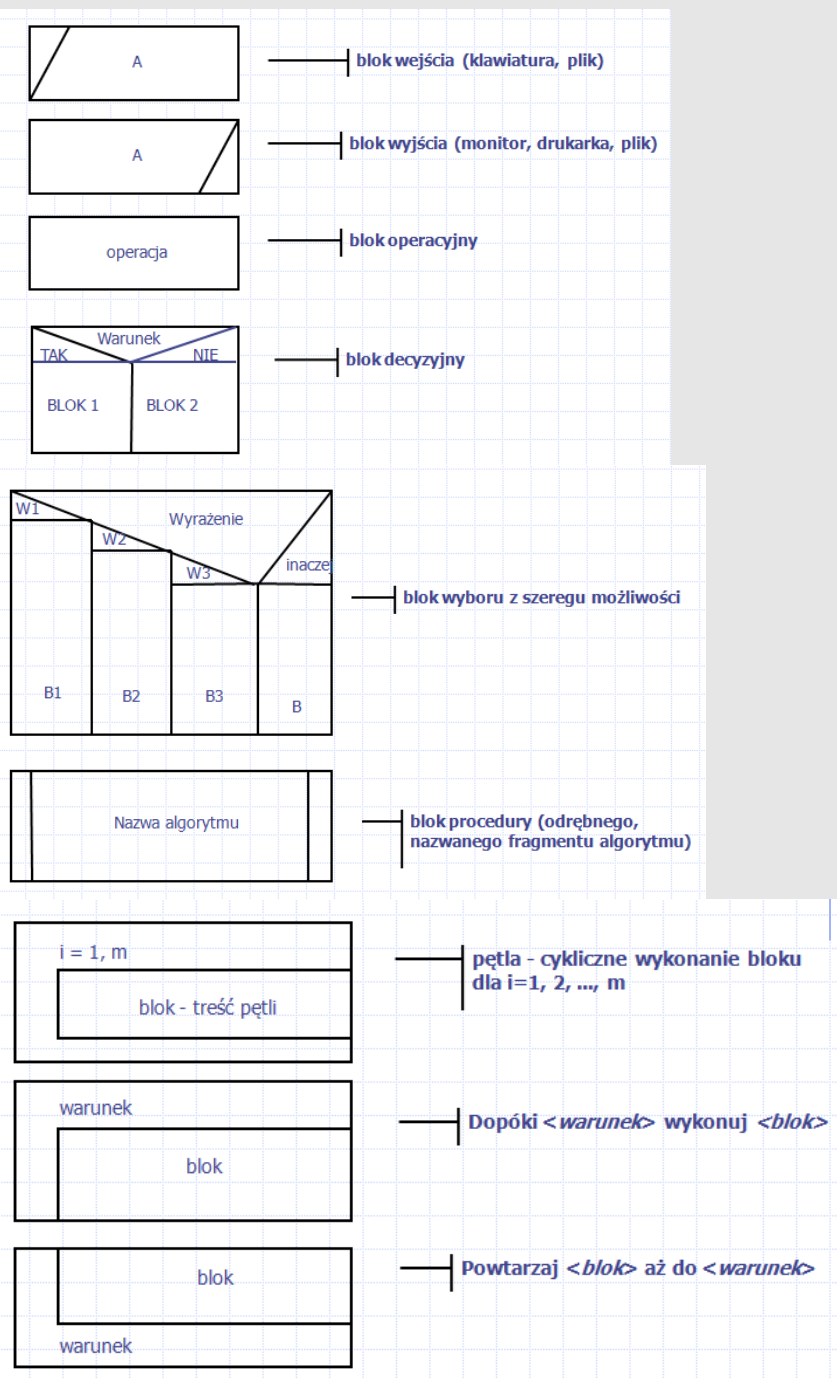
Pseudokod:

- zdanie proste
przypisz średniej wartość zero
czytaj x
pisz wynik
- zdanie decyzyjne
jeżeli warunek to zdanie
lub
jeżeli warunek to zdanie1
w przeciwnym przypadku
zdanie2
- zdanie wybierz
wybierz przełącznik z
wartość1: zdanie1
...
w innym przypadku zdanie_domyślne
- zdanie iteracyjne dopóki (podczas gdy)
dopóki warunek *wykonuj* zdanie
- zdanie iteracyjne powtarzaj
powtarzaj zdanie *aż do* warunek



- zdanie iteracyjne dla
 dla lista sytuacji **wykonuj** zdanie
- zdanie grupujące {...} lub begin ... end
 { //begin
 zdanie1
 zdanie2
 } //end

Schematy NS – Nassi - Schneidermana



Pytania kontrolne:

1. Podaj pojęcie i rolę algorytmu w programowaniu.
2. Wymień sposoby przedstawienia algorytmu.
3. Do jakich czynności (zdań pseudokodu) wykorzystywane są poszczególne bloki w schematach NS?

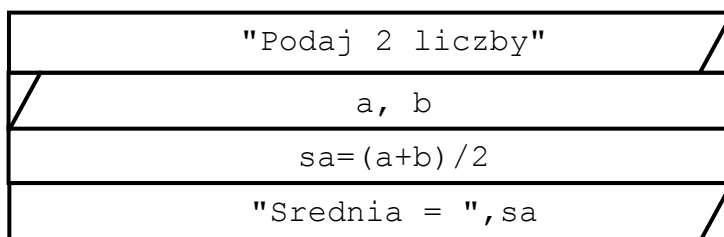
Zadania do analizy

Zadanie 2.1. Obliczanie średniej arytmetycznej dwóch liczb rzeczywistych (algorytm liniowy)

- Przeanalizuj poniższy algorytm w postaci listy kroków:

- | |
|--|
| 1. Wczytaj dwie liczby a i b. |
| 2. Dodaj do siebie te liczby i wynik podziel przez 2 $\rightarrow (a+b)/2$. |
| 3. Wyświetl wynik. Zakończ algorytm. |

- Przeanalizuj poniższy algorytm w postaci schematu NS:



Zadanie 2.2. Rozwiązywanie równania kwadratowego (algorytm z rozgałęzieniami)

Dane są współczynniki równania kwadratowego. Zbadać istnienie pierwiastków i jeśli istnieją obliczyć.

- Przeanalizuj poniższy algorytm w postaci listy kroków:

- | |
|---|
| 1. Wczytaj współczynniki równania: a, b, c ($a \neq 0$). |
| 2. Oblicz wyróżnik delta $\rightarrow d = b^2 - 4 * a * c$. |
| 3. Jeśli $d < 0$ wyświetl komunikat o braku pierwiastków. Zakończ algorytm. |
| 4. W przeciwnym wypadku:
Jeśli $d > 0$ to:
Oblicz pierwiastki $\rightarrow x_1 = (-b + \sqrt{d}) / (2 * a)$ $x_2 = (-b - \sqrt{d}) / (2 * a)$
Wyświetl wyniki. Zakończ algorytm. |
| 5. W przeciwnym wypadku ($d = 0$):
Oblicz pierwiastek $\rightarrow x = -b / (2 * a)$
Wyświetl wynik. Zakończ algorytm. |

- Przedstaw powyższy algorytm w postaci schematu NS.



Zadanie 2.3. Obliczanie średniej ocen studenta (algorytm iteracyjny)

Student w czasie sesji zimowej w PL zdaje n egzaminów. Obliczyć średnią sesji, jako zwykłą średnią arytmetyczną.

- Przeanalizuj poniższy algorytm w postaci listy kroków:
- Przedstaw poniższy algorytm w postaci schematu NS.

1. Wczytaj liczbę egzaminów n .
2. Wyzeruj zmienną $s \rightarrow s=0$.
3. Powtarzaj n razy:
Wczytaj kolejną ocenę x .
Dodaj ją do dotychczas obliczonej sumy $s \rightarrow s=s+x$.
4. Oblicz wartość średniej $\rightarrow s=s/n$.
5. Wyświetl wynik s . Zakończ algorytm.

Zadania do wykonania

Zadanie 2.4. Obliczanie pola powierzchni i objętości brył geometrycznych

Oblicz pole powierzchni i objętość sześcianu, prostopadłościanu, kuli i walca o podanych wymiarach.

- Przedstaw algorytm rozwiązania zadania w postaci schematu NS.

Zadanie 2.5. Policzenie zapotrzebowania na farbę do malowania pokoju

Ile puszek farby potrzeba na pomalowanie ścian pokoju o wymiarach $A \times B \times C$ (w metrach), jeśli 1 puszka (1 litr) wystarcza na pomalowanie $Y \text{ m}^2$? Okno w pokoju ma wymiary $1 \times 0,9$, drzwi: $2 \times 0,8$ (w metrach).

- Przedstaw algorytm rozwiązania zadania w postaci schematu NS.

Zadanie 2.6. Klasyfikacja wzrostu osoby (osób)

A. Określ, czy dana osoba jest niskiego, średniego czy wysokiego wzrostu (klasyfikację zastosuj wg własnego uznania).

- Przedstaw algorytm rozwiązania zadania w postaci schematu NS.

B. Dokonaj takiej klasyfikacji wzrostu dla N studentów.

- Przedstaw algorytm rozwiązania zadania w postaci schematu NS.

Zadanie 2.7. Wybranie najlepszej oceny w sesji studenta

Student w czasie sesji zimowej w PL zdaje n egzaminów.

A. Podaj najlepszą ocenę.

- Przedstaw algorytm rozwiązania zadania w postaci schematu NS.

B. Podaj, z którego egzaminu ocena była najlepsza.

- Przedstaw algorytm rozwiązania zadania w postaci schematu NS.

Zadanie 2.8. Obliczenie wypłat dla grupy pracowników

Wypłata dla pracownika składa się ze stawki bazowej, dodatku stażowego i premii. Dodatek stażowy przysługuje po 5 latach i wynosi:

$$\text{dodatek} = \begin{cases} 20\% & \text{gdy lata} > 20 \\ \text{tyle \% ile lat pracy} & \text{gdy lata} \in < 5, 20 > \end{cases}$$

Sporządzić listę wypłat dla N pracowników i sumaryczną wartość wypłat.

- Przedstaw algorytm rozwiązania zadania w postaci schematu NS.



LABORATORIUM 3. PROSTE PROGRAMY IMPERATYWNE. WPROWADZANIE I WYPROWADZANIE DANYCH. ZMIENNE RÓŻNYCH TYPÓW. INSTRUKCJA PRZYPISANIA. WYRAŻENIA.

Cel laboratorium:

Zaznajomienie z realizacją prostych algorytmów imperatywnych w zintegrowanym środowisku programistycznym Dev C++ (Code::Blocks).

Nabywanie praktycznych umiejętności pisania prostych programów z wykorzystaniem wyrażeń różnych typów.

Zakres tematyczny zajęć:

- idea programowania imperatywnego,
- struktura programu,
- stałe, zmienne i ich typy,
- wprowadzanie i wyprowadzanie danych,
- wyrażenia i operatory,
- tworzenie kodu źródłowego programu, kompilacja i wykonanie,
- polskie litery na ekranie konsoli.

Kompendium wiedzy:

Elementy programu:

- **identyfikatory** - ciąg znaków rozpoczynający się od litery lub _ (np. `x`, `X`, `sredniaOcen`, `srednia_ocen`),
- **słowa kluczowe** - nie mogą być identyfikatorem (np. `if`, `for`, `do`),
- **typy danych** podstawowe (`int`, `float`, `double`, `char`) i ich modyfikatory (`long`, `short`, `signed`, `unsigned`),
- **zmienne:**

<code>typ zmienna;</code>	np. <code>int x,y;</code>
<code>typ zmienna=wartość;</code>	np. <code>float f=0.25;</code>

- **stałe:**

```
#define nazwa wartość //dyrektywa preprocesora  
const typ nazwa=wartość;
```

- **Standardowe biblioteki ANSI C - dołączenie plików nagłówkowych:**

```
#include <stdio.h> //standardowa biblioteka WE/WY (np. printf(), scanf(), getchar(), .  
#include <stdlib.h> //funkcje ogólnego użytku (np. rand(), srand(),...)
```

- **Wyprowadzanie danych na ekran** – funkcje: `printf()`, `putchar()` z pliku nagłówkowego `<stdio.h>`



```
printf("łańcuch sterujący", argumenty);
putchar(znak);
```

- **Wczytywanie danych z klawiatury** – funkcje: `scanf()`, `getchar()` z pliku nagłówkowego `<stdio.h>`

```
scanf("łańcuch sterujący", argumenty);
getchar(znak);
```

```
char znak; int liczba_c; float liczba_rz; char imie[20];
printf("Podaj znak, liczbę całkowitą i liczbę
rzeczywistą\n");
scanf("%c %d %f", &znak, &liczba_c, &liczba_rz);
printf("Podajes znak: %c, liczbę całkowitą: %d, liczbę
rzeczywistą:%f\n",znak, liczba_c, liczba_rz );
printf("Podaj imię\n");
scanf("%s", imie);
printf("Witaj %s\n", imie);
```

- **Wyrażenie** to kombinacja operatorów i operandów (np. `suma=x+y;`).
- **Podstawowe typy operatorów:**
 - arytmetyczne: `+`, `-`, `*`, `/`, `%` (modulo – reszta z dzielenia), `++`, `--` (inkrementacja, dekrementacja),
 - przypisania: `=`, `+=`, `-=`, `*=`, `/=`,
 - relacyjne: `<`, `<=`, `>`, `>=`, `==` (równe), `!=` (różne),
 - logiczne: `&&` (koniunkcja I/AND), `||` (alternatywa LUB/OR), `!` (negacja NIE/NOT),
 - rzutowania: (typ).
- **Instrukcje** to wyrażenie zakończone średnikiem.

Podstawowe typy instrukcji:

- przypisania,
- warunkowa,
- wyboru,
- iteracyjne,
- wywołania funkcji.

- **Instrukcja przypisania**

```
zmienna=wartość;
zmienna+=wartość; //zmienna=zmienna+wartosc;
zmienna-=wartość; //zmienna=zmienna-wartosc;
zmienna*=wartość; //zmienna=zmienna*wartosc;
zmienna/=wartość; //zmienna=zmienna/wartosc;
```



Pytania kontrolne:

1. Na czym polega programowanie imperatywne?
2. Objaśnij pojęcie zmiennej i stałej.
3. Wymień znane ci podstawowe typy danych.
4. Objaśnij strukturę programu w języku C.
5. Jak zdefiniować stałe i zmienne w programie?
6. Podaj funkcje do wprowadzania i wyprowadzania danych.
7. Wymień formaty wprowadzania/wyprowadzania zmiennych różnych typów.
8. Objaśnij pojęcie wyrażenia w programowaniu. Wymień podstawowe operatory.
9. Podaj składnię i operatory instrukcji przypisania
10. Jak wyprowadzić na ekran teksty z polskimi literami?
11. Co to są kody ASCII?

Zadania do analizy

Zadanie 3.1. Struktura prostego programu imperatywnego w języku C

- Przeanalizuj przykład prostego programu imperatywnego (liniowy, sekwencyjny):

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #define PI 3.14159 //stale
4 #define AUTOR "Jan Kowalski"
5 #define KIERUNEK "Informatyka"
6 int main(int argc, char *argv[])
7 { //deklaracje
8   const int ROK=1; //stala
9   const float PREMIA=0.20;
10  float r, pole, stawka, wypлата;
11  int lg;
12  //instrukcje
13  printf("=====\n");
14  printf("*****\n");
15  printf("=====\n");
16  printf("Programowanie liniowe\n");
17  printf("Autor programu: %s, kierunek: %s, rok: %d\n",
18        AUTOR, KIERUNEK, ROK);
19  printf("=====\n");
20  printf("*****\n");
21  printf("=====\n");
22  printf("Podaj promień kola\n"); scanf("%f",&r);
23  pole=PI*r*r;
24  printf("Pole kola o promieniu %0.2f = %0.2f\n",
25        r,pole);
26  printf("=====\n");
27  printf("*****\n");
28  printf("=====\n");
```

```
29 printf("Pracownik1\n");
30 printf("Podaj liczbe godzin\n");
31 scanf("%d",&lg);
32 printf("Podaj stawke\n"); scanf("%f",&stawka);
33 wypłata= lg*stawka+ lg*stawka*PREMIA;
34 printf("Wypłata = %0.2f\n",wypłata);
35 printf("Pracownik2\n");
36 printf("Podaj liczbe godzin\n");
37 scanf("%d",&lg);
38 printf("Podaj stawke\n"); scanf("%f",&stawka);
39 wypłata= lg*stawka+ lg*stawka*PREMIA;
40 printf("Wypłata = %0.2f\n",wypłata);
41 printf("=====\n");
42 printf("*****\n");
43 printf("=====\n");
44 system("PAUSE");
45 return 0;
46 }
```

Zadanie 3.2. Polskie znaki narodowe

- Przeanalizuj program, pozwalający wyprowadzić na ekran konsoli polskie znaki narodowe. Zwróć uwagę na plik nagłówkowy `locale.h` i funkcję `setlocale` z parametrami `LC_ALL` i `""`.

```
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
int main(int argc, char *argv[]) {
    setlocale(LC_ALL, "");
    printf("ąęółźż");
    return 0;
}
```

Zadania do wykonania

Zadanie 3.3. Dane studenta

Napisz program, który na podstawie wprowadzonych z klawiatury danych (imię, nazwisko, wiek, płeć) i zdefiniowanych stałych (`STATUS` przyjmujący wartość `student` i `SREDNIA` przyjmująca wartość twojej pożądanej średniej ocen - liczba rzeczywista) wyświetli w jednym wierszu imię, nazwisko, i płeć, a w drugim twój status i średnią.



Zadanie 3.4. Obliczanie objętości i pola powierzchni brył

Napisz program obliczający objętość i pole powierzchni sześcianu, prostopadłościanu o podstawie kwadratowej i prostokątnej oraz walca.

Zadanie 3.5. Obliczanie średniej arytmetycznej i średniej geometrycznej dwóch liczb całkowitych

Napisz program obliczający średnią arytmetyczną i średnią geometryczną dwóch liczb całkowitych.

Zadanie 3.6. Przeliczanie wielkości fizycznych

Napisz program przeliczający wielkości fizyczne:

- z mili na kilometry (1 mila=0.625 km),
- z kilometrów na mile (1 km=1.6 mili),
- z koni mechanicznych [KM] na waty [W] (1KM = 735W)),
- z kilometrów na godzinę [km/h] na metry na sekundę [m/s],
- ze stopni Fahrenheita [°F] na stopnie Celsjusza [°C] ($t^{\circ}\text{C}=5/9(t^{\circ}\text{F}-32)$).

Zadanie 3.7. Wiek w przyszłości

Napisz program, który na podstawie twojego obecnego wieku (w latach i miesiącach) i podanego okresu czasu (w miesiącach), obliczy twój wiek w przyszłości (w latach i miesiącach).

Zadania dodatkowe

Zadanie 3.8. Wyrażenia

Napisz program obliczający wartość wyrażeń (wykorzystaj funkcje standardowe):

a) $10\cos x - 0,1x^2 + \sin x + \sqrt{4x^2 + 7}$

b) $\lg(x+5) + e^{x+1} - |tgx + 1|$

c) $\frac{\sin^2 \alpha + 0,5}{\cos \alpha^4 + tg^4 \alpha^2}$

d) $\sqrt{\frac{|5\sin \beta^5 + 1|}{3,5(\sin \beta + \cos \beta)^2}}$



Zadanie 3.9. Wyrażenia - funkcje trygonometryczne

Napisz program obliczający wartość funkcji trygonometrycznych: $\sin\alpha$, $\cos\alpha$ i $\tan\alpha$ dla α podanego w stopniach np. 90° , 120° , 180° .

Zadanie 3.10. Zamiana miejscami zmiennych i liczba odwrotna

Zmienne a i b to dwie liczby całkowite trzycyfrowe. Napisz program zamieniający miejscami wartości tych zmiennych. Po zamianie wyświetl te liczby, a następnie zmodyfikuj je zamieniając miejscami cyfrę jedności i cyfrę setek - wyświetl liczby w odwrotnej kolejności cyfr.

Zadanie 3.11. Obwód okręgu

Napisz program, który obliczy obwód okręgu, który przechodzi przez punkt A(x1, y1) i którego środek znajduje się w punkcie B(x2, y2).

Zadanie 3.12. Znaki

Napisz program, który:

- a) po podaniu dowolnego znaku wyświetli go wraz z kodem ASCII, a następnie wyświetli znak o kodzie następnym,
- b) po podaniu małej litery zamieni ją na dużą.



LABORATORIUM 4. PROSTE PROGRAMY STRUKTURALNE. FUNKCJE STANDARDOWE I FUNKCJE WŁASNE. ARGUMENTY FUNKCJI.

Cel laboratorium:

Zaznajomienie z realizacją algorytmów liniowych (sekwencyjnych), z ideą powtórnego wykorzystania kodu i programowaniem strukturalnym. Nabycie praktycznych umiejętności pisania prostych programów strukturalnych podzielonych na mniejsze bloki z wykorzystaniem funkcji własnych i standardowych.

Zakres tematyczny zajęć:

- programowanie strukturalne, proceduralne, modułowe
- wykorzystanie funkcji standardowych,
- prototypy, definicje i wywołania funkcji własnych,
- parametry funkcji.

Kompendium wiedzy:

Programowanie strukturalne wykorzystuje ideę dzielenia programu na mniejsze części – bloki typu: sekwencja, wybór, iteracja z jednym wejściem i jednym lub kilkoma wyjściami. Jeżeli blokiem jest funkcja, umieszczona w pliku z funkcją główną main, może być ona wielokrotnie wykorzystana w tym samym programie – **programowanie proceduralne**. Większym blokiem do wielokrotnego wykorzystania w wielu programach jest moduł – plik źródłowy lub nagłówkowy dołączony do programu – **programowanie modułowe (modularne)**.

Funkcja – wydzielony fragment kodu spełniający określone czynności. Może zwracać wartość określonego typu (instrukcja **return**) lub nie (typ **void**).

Prototyp funkcji – zapowiedź funkcji:

```
typ_wyniku nazwa_funkcji();  
typ_wyniku nazwa_funkcji(parametry_formalne);
```

Definicja funkcji – opis algorytmu funkcji:

```
typ_wyniku nazwa_funkcji()  
{//deklaracje i instrukcje;  
    return wyrażenie; //opcjonalnie  
}  
typ_wyniku nazwa_funkcji(parametry_formalne)  
{//deklaracje i instrukcje;  
    return wyrażenie; //opcjonalnie  
}
```

Wywołanie funkcji:

```
nazwa_funkcji();  
nazwa_funkcji(parametry_aktualne);
```



W programie można wykorzystywać funkcje własne i funkcje standardowe, zgrupowane w odpowiednich modułach – dyrektywą **#include** należy dołączyć wybrany plik nagłówkowy.

Wykorzystanie funkcji standardowych:

```
#include <plik_naglowkowy_z_funkcjami>
```

```
nazwa_funkcji();
```

```
nazwa_funkcji(parametry_aktualne);
```

Grupy wybranych funkcji standardowych:

- Funkcje matematyczne <math.h>
 **asin(x), acos(x), atan(x), sin(x), cos(x), tan(x),
 exp(x), log(x), pow(x,y), sqrt(x), ceil(x), floor(x),
 fabs(x), fmod(x,y), ...**
- Funkcje łańcuchowe <string.h>
 strlen(), strcat(), strcmp(), strcpy(), ...
- Funkcje znakowe <ctype.h>
 tolower(), toupper(), isalpha(), isdigit(), isalnum(),
- Funkcje ogólnego użytku <stdlib.h>
 abs(), rand(), qsort(), ...

Prosty program proceduralny w języku C posiada następującą strukturę:

```
//dyrektywy preprocesora  
  
//prototypy funkcji własnych  
  
int main() //naglowek funkcji głównej  
{  
    //deklaracje  
    //instrukcje korzystające z ww. funkcji  
    return 0; //instrukcja powrotu  
}  
  
//definicje funkcji własnych
```

Pytania kontrolne:

1. Na czym polega programowanie strukturalne?
2. Na czym polega programowanie proceduralne?
3. Objaśnij pojęcie funkcji.
4. Jak wykorzystać w programie funkcje standardowe?
5. Co to jest prototyp funkcji?
6. Jak zdefiniować funkcję?
7. Jak wywołać funkcję?
8. Objaśnij pojęcia: parametry formalne i parametry aktualne funkcji.
9. Jak wygląda struktura programu pisana techniką (paradygmatem) programowania proceduralnego?



Zadania do analizy**Zadanie 4.1. Programowanie proceduralne - struktura programu w języku C**

- Przeanalizuj przykład programu pisanego techniką proceduralną i porównaj go z kodem źródłowym zadania 2.2:

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <math.h> //M_PI
4
5  #define AUTOR "Jan Kowalski"
6  #define KIERUNEK "Informatyka"
7
8  //prototypy funkcji=====
9  void szlaczek(); //funkcja bez parametrów
10 void info(); //funkcja bez parametrów
11 float pole(float promien); //funkcja z 1 parametrem
12 float wyplata(int godziny, float stawka); //funkcja z 2 par.

13 int main(int argc, char *argv[]) //funkcja glowna=====
14 { //deklaracje
15     float r,st;
16     int lg;
17     //instrukcje
18     szlaczek(); //wywołanie funkcji
19     info();
20     szlaczek();
21     printf("Podaj promien kola "); scanf("%f",&r);
22     printf("Pole kola o promieniu %0.2f= %0.2f\n",r, pole(r));
23                                     //wywołanie funkcji
24     szlaczek();
25     printf("Pracownik1\n");
26     printf("Podaj liczbe godzin "); scanf("%d",&lg);
27     printf("Podaj stawke "); scanf("%f",&st);
28     printf("Wyplata = %0.2f\n", wyplata(lg,st));
29                                     //wywołanie funkcji
30     printf("Pracownik2\n");
31     printf("Podaj liczbe godzin "); scanf("%d",&lg);
32     printf("Podaj stawke "); scanf("%f",&st);
33     printf("Wyplata = %0.2f\n",wyplata(lg,st));
34                                     //wywołanie funkcji
35     szlaczek();
36     system("PAUSE");
37     return 0;
38 }
39 //definicje funkcji=====
40 void szlaczek()
41 {

```



```
42 printf("=====\n");
43 printf("*****\n");
44 printf("=====\n");
45 }//=====
46 void info()
47 { const int ROK=1;
48   printf("Programowanie liniowe\n");
49   printf("Autor programu: %s kierunek: %s rok:
50         %d\n", AUTOR, KIERUNEK, ROK);
51 } //=====
52 float pole(float promien)
53 {
54   return M_PI*pow(promien,2);
55 }//=====
56 float wypłata(int godziny, float stawka)
57 { const float PREMIA=0.20;
58   return godziny*stawka+ godziny*stawka*PREMIA;
59 }//=====
```

Zadanie 4.2. Programowanie proceduralne - struktura programu w języku C

- Przeanalizuj przykład programu pisanego techniką proceduralną.
- Podaj tekst w komentarzach.

```
1  #include <stdio.h> //???
2  #include <stdlib.h> //???
3  //ponizej sa ???
4  int suma(int a, int b);
5  int roznica(int a, int b);
6  int iloczyn(int a, int b);
7  float iloraz(int a, int b);
8  void iloraz2(int a, int b);
9
10 int main(int argc, char *argv[])
11 {int x,y,reszta;
12 printf("Podaj 2 liczby calkowite\n");//???
13 scanf("%d %d", &x, &y);//???
14 printf("suma: %d \n", suma(x,y));
15 printf("roznica: %d \n", roznica(x,y)); //???
16 printf("iloczyn: %d \n", iloczyn(x,y));
17 printf("iloraz - wynik dzielenia rzeczywistego:
18        %0.2f\n", iloraz(x,y));
19 iloraz2(x,y);
20 system("PAUSE");
21 return 0;
22 }
23 // ponizej sa???
24 int suma(int a, int b)
```



```
25 { return a+b; }

26 int roznica(int a, int b)
27 { return a-b;}
28 int iloczyn(int a, int b)
29 { return a*b;}

30 float iloraz(int a, int b)
31 { return (float)a/b; }

32 void iloraz2(int a, int b)
33 { printf("iloraz - wynik dzielenia calkowitego:
34         %d reszta: %d\n",a/b, a%b);
35 }
```

Zadania do wykonania

Zrealizuj zadania do wykonania z laboratorium 2 techniką programowania proceduralnego:

Zadanie 4.3. Dane studenta

Napisz funkcję, która na podstawie wprowadzonych z klawiatury danych (imię, nazwisko, wiek, płeć) i zdefiniowanych stałych (STATUS przyjmujący wartość student i SREDNIA przyjmująca wartość twojej pożądanej średniej ocen - liczba rzeczywista) wyświetli w jednym wierszu imię, nazwisko, i płeć, a w drugim twój status i średnią. Wywołaj tą funkcję.

Zadanie 4.4. Obliczanie objętości i pola powierzchni brył.

Napisz funkcje obliczające objętość i pole powierzchni sześcianu, prostopadłościanu o podstawie kwadratowej i prostokątnej oraz walca. Wywołaj te funkcje.

Zadanie 4.5. Obliczanie średniej arytmetycznej i średniej geometrycznej dwóch liczb całkowitych

Napisz funkcje obliczające średnią arytmetyczną i średnią geometryczną dwóch liczb całkowitych. Wywołaj te funkcje.

Zadanie 4.6. Przeliczanie wielkości fizycznych

Napisz funkcje przeliczające wielkości fizyczne:

- z mili na kilometry (1 mila=0.625 km),
- z kilometrów na mile (1 km=1.6 mili),
- z koni mechanicznych [KM] na waty [W] (1KM = 735W)),



- z kilometrów na godzinę [km/h] na metry na sekundę [m/s],
- ze stopni Fahrenheita [$^{\circ}\text{F}$] na stopnie Celsjusza [$^{\circ}\text{C}$] ($t^{\circ}\text{C}=5/9(t^{\circ}\text{F}-32)$).

Zadanie 4.7. Wiek w przyszłości

Napisz funkcję, która na podstawie twojego obecnego wieku (w latach i miesiącach) i podanego okresu czasu (w miesiącach), obliczy twój wiek w przyszłości (w latach i miesiącach). Wywołaj tę funkcję.

Zadania dodatkowe

Zadanie 4.8. Wyrażenia

Napisz i wywołaj funkcje obliczające wartość wyrażeń (wykorzystaj funkcje standardowe):

e) $10\cos x - 0,1x^2 + \sin x + \sqrt{4x^2 + 7}$

f) $\lg(x+5) + e^{x+1} - |tgx+1|$

g) $\frac{\sin^2 \alpha + 0,5}{\cos \alpha^4 + tg^4 \alpha^2}$

h) $\sqrt{\frac{|5\sin \beta^5 + 1|}{3,5(\sin \beta + \cos \beta)^2}}$

Zadanie 4.9. Wyrażenia - funkcje trygonometryczne

Napisz i wywołaj funkcję obliczającą wartość funkcji trygonometrycznych $\sin \alpha$, $\cos \alpha$ i $tg \alpha$ dla α podanego w stopniach np. 90° , 120° , 180° .

Zadanie 4.10. Zamiana miejscami zmiennych i liczba odwrotna

Zmienne a i b to dwie liczby całkowite trzycyfrowe. Napisz i wywołaj funkcję zamieniającą miejscami wartości tych zmiennych. Po zamianie wyświetl te liczby, a następnie zmodyfikuj je zamieniając miejscami cyfrę jedności i cyfrę setek - wyświetl liczby w odwrotnej kolejności cyfr.

Zadanie 4.11. Obwód okręgu

Napisz i wywołaj funkcję, która obliczy obwód okręgu, który przechodzi przez punkt A(x1, y1) i którego środek znajduje się w punkcie B(x2, y2).



Zadanie 4.12. Znaki

Napisz i wywołaj funkcje, które:

- a) po podaniu dowolnego znaku wyświetli go wraz z kodem ASCII, a następnie wyświetli znak o kodzie następnym,
- b) po podaniu małej litery zamieni ją na dużą.



Fundusze Europejskie
Wiedza Edukacja Rozwój



**Rzeczpospolita
Polska**

Unia Europejska
Europejski Fundusz Społeczny



LABORATORIUM 5. INSTRUKCJE WARUNKOWE IF, IF...ELSE. OPERATOR WARUNKOWY. INSTRUKCJA WYBORU SWITCH. INSTRUKCJA BREAK.

Cel laboratorium:

Zaznajomienie z realizacją algorytmów z rozgałęzieniami z wykorzystaniem instrukcji warunkowej lub z wykorzystaniem operatora ternarnego (warunkowego) oraz z wykorzystaniem instrukcji wyboru. Nabycie praktycznych umiejętności programowania algorytmów z rozgałęzieniami.

Zakres tematyczny zajęć:

- instrukcje warunkowe,
- operator warunkowy,
- instrukcja złożona (grupująca, blokowa),
- instrukcja wyboru,
- instrukcja przerwania BREAK.

Kompendium wiedzy:

Instrukcje warunkowe służą do sterowania przebiegiem programu w zależności od spełnienia lub nie spełnienia określonego warunku. Umożliwiają programową realizację algorytmów z rozgałęzieniami. Instrukcje warunkowe mogą występować w następujących postaciach:

```
if (wyrażenie) instrukcja;  
// Jeżeli wyrażenie jest prawdziwe (≠0) wykonuj instrukcję  
if (wyrażenie) instrukcja1;  
    else instrukcja2;  
//Jeżeli wyrażenie jest prawdziwe (≠0) wykonuj instrukcję1,  
// w przeciwnym wypadku wykonuj instrukcję2
```

np. `if(a>b) max=a; if(a>b) max=a; else max=b;`

Jeżeli w zależności od wartości wyrażenia ma być wykonywanych wiele instrukcji należy użyć instrukcji złożonej (grupującej) `{}`:

```
if (wyrażenie) {instrukcja1;  
                instrukcja2;  
            } /*instrukcja złożona (grupująca)*/  
else {instrukcja3;  
      instrukcja4;  
      } /*instrukcja złożona (grupująca)*/  
//Jeżeli wyrażenie jest prawdziwe (≠0) wykonuj blok instrukcji (np. instrukcję1 i 2  
//w przeciwnym wypadku wykonuj blok instrukcji (np. instrukcję3 i 4).
```

```
np. if(a>b){max=a; printf("Max z 2 liczb =%d\n",max);}  
    else{max=b; printf("Max z 2 liczb =%d\n",max);}
```



Instrukcje warunkowe mogą być zagnieżdżone (jedna w drugiej), przy czym obowiązuje zasada, że `else` związane jest z najbliższym `if`:

```
np.    if(a>b) printf("Liczba a jest większa\n");
        else if(a<b) printf("Liczba b jest większa\n");
        else printf("Liczby a i b są równe\n");
```

Wyrażenie po słowie `if` może zawierać następujące operatory porównania:

`==, !=, <, >, <=, >=`.

Wyrażenie może również zawierać operatory logiczne: `&&`, `||`, `!`.

Alternatywą dla instrukcji warunkowej jest wykorzystanie operatora warunkowego.

Operator ternarny (trójoperandowy operator warunkowy) wykorzystuje trzy operandy, z których każdy jest wyrażeniem:

```
wyrażenie1 ? wyrażenie2 : wyrażenie3;
//Jeżeli wyrażenie1 jest prawdziwe (≠0) to całe wyrażenie przyjmuje wartość
//wyrażenia2, w przeciwnym wypadku wyrażenia3
```

```
np. max=(a>b) ? a : b;
```

Instrukcja wyboru może występować w następujących postaciach:

- warianty wyboru zakończone instrukcją `BREAK`:

```
switch (wyrażenie typu int, char lub enum)
{
    case stała1: ciąg_instrukcji_1; break;
    case stała2: ciąg_instrukcji_2; break;
    ...
    default://opcjonalnie
        ciąg_instrukcji_D;
}
// Jeżeli wyrażenie przyjmuje wartość stałej1 wykonuj ciąg_instrukcji_1 i wyjdź ze switch
//Jeżeli wyrażenie przyjmuje wartość stałej2 wykonuj ciąg_instrukcji_2 i wyjdź ze switch
//... w przeciwnym wypadku wykonuj ciąg_instrukcji_D.
```

```
np. switch (ocena) {
    case 2: printf("ndst");break;
    case 3: printf("dst"); break;
    case 4: printf("db"); break;
    case 5: printf("bdb");break;
    default: printf("zla ocena");
}
```

- warianty wyboru z opuszczoną instrukcją `BREAK`:

```
switch (wyrażenie typu int, char lub enum)
{
    case stała1:
    case stała2: ciąg_instrukcji_2; break;
    ...
    default://opcjonalnie
        ciąg_instrukcji_D;
}
// Jeżeli wyrażenie przyjmuje wartość stałej1 lub stałej2 wykonuj ciąg_instrukcji_2 i wyjdź
//...w przeciwnym wypadku wykonuj ciąg_instrukcji_D.
```



```
np. switch (ocena) {
    case 2: printf("negatywna ocena");break;
    case 3:
    case 4:
    case 5: printf("pozytywna ocena");break;
    default: printf("zla ocena");
}
```

Zwiększenie czytelności programu można uzyskać stosując **typy wyliczeniowe** w instrukcjach sterujących – zbiór czytelnych nazw o wartościach stałych całkowitych np.:

```
enum kolory {czerwony, pomaranczowy, zolty, zielony,
niebieski, blekitny, fioletowy};
```

```
//typ wyliczeniowy o wartościach: 0, 1, 2, ..6
```

```
enum kolory kol; //zmienna wyliczeniowego typu
```

Pytania kontrolne:

1. Podaj dwie alternatywne składnie instrukcji warunkowej.
2. Podaj dowolne przykłady instrukcji warunkowej z jednym warunkiem oraz z kilkoma warunkami.
3. Jak interpretowana jest wartość logiczna PRAWDA w warunku instrukcji warunkowej, a jak wartość FAŁSZ?
4. Podaj postać i zastosowanie operatora warunkowego.
5. Podaj postać i zastosowanie instrukcji złożonej (grupującej, blokowej).
6. Podaj składnię i zastosowanie instrukcji wyboru.
7. Podaj zastosowanie instrukcji BREAK w instrukcji wyboru.
8. Wymień ograniczenia instrukcji SWITCH w stosunku do instrukcji IF.

Zadania do analizy

Zadanie 5.1. Instrukcje warunkowe i operator warunkowy

- Przeanalizuj przykład programu wykorzystującego instrukcje warunkowe i operator warunkowy.
- Podaj tekst w komentarzach.

```
1  #include <stdio.h>  //???
2  #include <stdlib.h>
3
4  float f(float x,float y);  //???
5  void punkt(float x, float y);  //???
6  char parzysta(int a);  //???
7
8  int main(int argc, char *argv[])
9  {float xx,yy; int liczba;
10  printf("Podaj x=");
11  scanf("%f",&xx);
12  printf("Podaj y=");
13  scanf("%f",&yy);
```



```
14 printf("wynik funkcji=%0.2f\n", f(xx,yy)); //???
15 punkt(xx,yy); //???
16 printf("Podaj liczbe calkowita");
17 scanf("%d",&liczba);
18 printf("Liczba %d parzysta: %c\n",liczba,
    parzysta(liczba));
19
20 system("PAUSE");
21 return 0;
22 }
23
24 float f(float x,float y) //???
25 {if(x<0 && y<0) return (x*x + y*y);
26   else if(x*y <= 0) return 0;
27   else return sqrt(x+y);
28 }
29
30 void punkt(float x, float y) //???
31 {if (x*y ==0) printf("punkt na osi\n");
32   else if(x>0)
33       if (y>0) printf("punkt w I cw.\n");
34       else printf("punkt w IV cw.\n");
35   else if(y>0) printf("punkt w II cw.\n");
36   else printf("punkt w III cw.\n");
37 }
38
39 char parzysta(int a) //???
40 { return (a%2==0)?'T':'N'; //???
41 // if(a%2==0) return 'T'; else return 'N'; //???
42 }
```

Zadanie 5.2. Instrukcja wyboru i instrukcja BREAK

- Przeanalizuj przykład programu wykorzystującego instrukcje SWITCH i BREAK.
- Podaj tekst w komentarzach.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  enum kolory{czerwony, pomaranczowy, zolty,
5  zielony, niebieski, blekitny, fioletowy}; //???
6  void kalkulator(double x, double y,char znak); //???
7  void samogloska(char litera); //???
8  void tecza(enum kolory kol); //???
9  //=====
10 int main(int argc, char *argv[]) //???
11 { double a,b;
12 char zn,l; //???
```



```

13 int k, wybor;
14 printf("MENU PROGRAMU:\n");
15 printf("1. Kalkulator\n");
16 printf("2. Samogloska\n");
17 printf("3. Kolory teczny\n");
18 printf("Wybierz pozycje\n");
19 scanf("%d",&wybor);
20 switch(wybor) //???
21 { case 1: //???
22     printf("podaj 2 liczby i znak dzialania\n
23         ");
24     scanf("%lf %lf %c", &a,&b, &zn);
25     kalkulator(a,b,zn); break; //???
26 case 2:
27     printf("podaj litere\n");
28     getche(); //scanf("%c", &l);
29     printf("\n");
30     samogloska(l); break; //???
31 case 3:
32     printf("Wybierz pozycje koloru w teczy od
33         0 do 6\n");
34     scanf("%d", &k);
35     teczka(k); break; //???
36 default:
37     printf("zly wybor\n"); //???
38 }
39 system("PAUSE");
40 return 0;
41 }
42 //=====
43 void kalkulator(double x, double y,char znak) //???
44 {
45     switch (znak) { //???
46     case '+': printf("suma= %0.2lf\n",x+y);
47                 break;
48     case '-': printf("roznica= %0.2lf\n",
49                 x-y);break;
50     case '*': printf("iloczyn= %0.2lf\n",x*y);break;
51     case '/': if (y) {printf("iloraz=
52                 %0.2lf\n",x/y);}
53     else printf("y=0\n");break;
54 default: printf("zly znak\n\n");
55 }
56 }
57 //=====
58 void samogloska(char litera) //???
59 { //nie uwzgledniono polskich liter a, e o
60 switch(litera) //???
61 { case 'a':

```



```
57 case 'A':
58 case 'e':
59 case 'E':
60 case 'i':
61 case 'I':
62 case 'o':
63 case 'O':
64 case 'u':
65 case 'U':
66 case 'y':
67 case 'Y': printf("samogloska\n"); break;
68 default: printf("nie samogloska\n");
69 }}
70 //=====
71 void tecza(enum kolory kol) //???
72 {
73 switch (kol){
74 case czerwony: printf("czerwony ");break; //???
75 case pomaranczowy: printf("pomaranczowy ");break;
76 case zolty: printf("zolty "); break;
77 case zielony: printf("zielony ");break;
78 case niebieski: printf("niebieski ");break;
79 case blekitny: printf("blekitny ");break;
80 case fioletowy: printf("fioletowy"); break;
81 default: printf("brak takiego koloru\n");
82 }
83 switch (kol){ //???
84 case czerwony:
85 case pomaranczowy:
86 case zolty: printf(" - wybrano cieply
87               kolor\n"); break;
88 case zielony:
89 case niebieski:
90 case blekitny:
91 case fioletowy: printf(" - wybrano chlodny
92               kolor\n");
93 } //=====
```

Zadania do wykonania

Zadanie 5.3. Obliczanie wartości funkcji

Napisz funkcję1 obliczającą wartość z wykorzystując instrukcję warunkową.

Napisz funkcję2 obliczającą wartość z wykorzystując instrukcję wyboru.

Wywołaj te funkcje.



Fundusze Europejskie
Wiedza Edukacja Rozwój



**Rzeczpospolita
Polska**

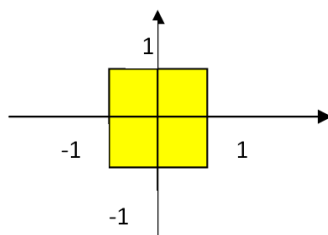
Unia Europejska
Europejski Fundusz Społeczny



$$z = \begin{cases} 1 - \sin \alpha & t = 8 \\ \frac{1}{2}(1 + \cos \alpha) & t = 0, 1, 2, 3 \\ \sqrt{\alpha^2 + 1} & t = 4, 6, 7 \end{cases}$$

Zadanie 5.4. Przynależność punktu do wskazanego obszaru

Napisz funkcję sprawdzającą, czy punkt o współrzędnych x, y należy do zamalowanego obszaru. Wywołaj tę funkcję.



Zadanie 5.5. Równanie kwadratowe

Napisz funkcję obliczającą pierwiastki równania kwadratowego $ax^2+bx+c=0$ uwzględniającą wszelkie możliwe warianty danych a, b, c . Wywołaj tę funkcję.

Zadanie 5.6. Pole trójkąta

Napisz i wywołaj funkcję, która na podstawie 3 liczb - boków trójkąta obliczy jego pole. Zweryfikuj czy podane liczby utworzą trójkąt.

Zadanie 5.7. Miesiące

Napisz funkcję, która na podstawie numeru miesiąca określi, do jakiego kwartału roku on należy i ile dni zawiera. Wywołaj tę funkcję.

Zadania dodatkowe

Zadanie 5.8. Szczęśliwy bilet

Bilet tramwajowy posiada sześciocyfrowy numer. Napisz funkcję sprawdzającą, czy jest to bilet „szczęśliwy”. Bilet uznawany jest za „szczęśliwy”, jeżeli suma 3 pierwszych i 3 ostatnich cyfr jest taka sama. Wywołaj tę funkcję.



Zadanie 5.9. Wypłata pracownika

Napisz funkcję, która na podstawie pensji i stażu pracownika obliczy jego wypłatę w następujący sposób: jeżeli staż pracownika jest mniejszy niż 5 lat, dodatek stażowy się nie należy, jeżeli pracownik przepracował w firmie od 5 do 10 lat, dodatek stażowy wynosi tyle procent ile lat ma staż pracownika, jeżeli staż pracownika jest powyżej 10 lat, dodatek stażowy wynosi 15%. Wywołaj tę funkcję.

Zadanie 5.10. Stypendium studenta

Napisz funkcję, która na podstawie 3 ocen z egzaminów wyświetli informację o przyznanym (lub nie) stypendium. Student otrzyma stypendium, jeśli zdał wszystkie egzaminy. Jeśli średnia ocen jest większa od 4 otrzyma stypendium 500 zł; jeśli $3 < \text{średnia} \leq 4$ to stypendium wynosi 300 zł. Wywołaj tę funkcję.

Zadanie 5.11. Liczby

Dane są trzy dodatnie liczby całkowite a, b, c. Jeżeli wszystkie są parzyste oblicz ich sumę, jeżeli dowolna z nich to 1 oblicz ich iloczyn, w pozostałych przypadkach zwróć -1. Zdefiniuj i wywołaj odpowiednią funkcję.

Zadanie 5.12. Znaki

Napisz i wywołaj funkcję, która sprawdzi, czy podany z klawiatury znak jest:

- znakiem dolara,
- małą literą angielskiego alfabetu,
- dużą literą angielskiego alfabetu,
- cyfrą,
- znakiem działania podstawowych operacji arytmetycznych.



LABORATORIUM 6. INSTRUKCJE ITERACYJNE WHILE, DO...WHILE. INSTRUKCJA CONTINUE.

Cel laboratorium:

Zaznajomienie z realizacją algorytmów iteracyjnych. Nabycie praktycznych umiejętności programowania algorytmów iteracyjnych z wykorzystaniem instrukcji WHILE, DO...WHILE.

Zakres tematyczny zajęć:

- pojęcie i organizacja iteracji,
- instrukcje pętli z warunkiem wejścia WHILE, oraz pętli z warunkiem wyjścia DO...WHILE,
- zastosowanie instrukcji BREAK i CONTINUE w iteracjach.

Kompendium wiedzy:

Instrukcje iteracyjne (pętli): **while** i **do while** służą do powtarzania fragmentu programu, dopóki wartość wyrażenia wpisanego w instrukcję pętli jest różna od zera, czyli wyrażenie jest prawdziwe. W pętli **do...while** sprawdzenie wyrażenia następuje po pierwszym wykonaniu instrukcji, a więc taka pętla wykona się co najmniej raz.

```
while (wyrażenie) instrukcja;  
//dopóki wyrażenie jest prawdziwe wykonuj instrukcję  
  
while (wyrażenie)          //użycie instrukcji złożonej w while  
{instrukcja1;  
  instrukcja2;  
  .....  
}  
do {instrukcja} while (wyrażenie);  
//wykonuj instrukcję dopóki wyrażenie jest prawdziwe
```

Instrukcja **continue** powoduje wykonanie kolejnej iteracji (jeśli wyrażenie sterujące powtórzeniem jest prawdziwe) w bloku **while** (lub **do...while**) z pominięciem instrukcji następujących po **continue**.

```
np. int i=2; while (i<10) i=i*i;           //p1  
  
int i=2; do {i=i*i;} while (i<10);        //p2  
int i=0; while (i<6)                       //p3  
{ i++;  
  if (i==3) continue;  
  printf("%d\n",i);  
}
```



Pytania kontrolne:

1. Objaśnij pojęcie iteracji.
2. Podaj składnię i zastosowanie instrukcji WHILE.
3. Podaj składnię i zastosowanie instrukcji DO ...WHILE.
4. Podaj zastosowanie instrukcji BREAK i CONTINUE w iteracjach.

Zadania do analizy

Zadanie 6.1. Instrukcje iteracyjne

- Przeanalizuj przykład programu wykorzystującego instrukcje iteracyjne.

Funkcja **suma** oblicza sumę podanych liczb, jeżeli są >100 .

Funkcja **tablicaWartosci** oblicza i wyświetla wartości wyrażenia $x + \sin x$ dla $x \in \langle a, b \rangle$, z krokiem k .

Funkcja **iloczyn** oblicza iloczyn liczb z przedziału $\langle a, b \rangle$.

Funkcja **zagadka** realizuje najwyżej pięciokrotne zgadywanie przez użytkownika wylosowanej liczby całkowitej jednocyfrowej. Zwraca liczbę prób do uzyskania sukcesu lub -1, jeżeli użytkownik nie zgadł.

- Podaj tekst w komentarzach.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  float suma(); //???
5  void tablicaWartosci(float a, float b, float k); //???
6  int iloczyn(int a, int b); //???
7  int zagadka (int a); //???
8
9  int main ()
10 {int nr=1;
11     while (nr) //???
12     {printf ("Wpisz numer funkcji (1, 2, 3 lub 4)\n,
13         Koniec programu - wpisz 0  ");
14     scanf( "%d", &nr);
15     if (nr==1) printf( "suma liczb = %f", suma()); //???
16     else
17     if(nr==2)
18     {    float od,do,krok;
19         printf("Podaj konce przedzialu i krok\n");
20         scanf("%f%f%f", &od, &do, &krok);
21         tablica_wartosci(od, do, krok); //???
22     }
23     else
24     if(nr==3)
25     {    int od,do;
```



```
25         printf("Podaj konce przedzialu \n");
26         scanf("%d%d", &od, &do);
27         printf(" iloczyn liczb podzielnych przez 3
28         =%d\n", iloczyn(od, do)); //???
29     }
30     else
31     {
32         if(nr==4)
33         {
34             int a,b;
35             a=rand()%10;
36             b=zagadka(a); //???
37             if (b== -1) printf(" nie zgadles\n");
38             else printf("zgadles za %d razem\n",b);
39         }
40     }
41     printf("\n KONIEC\n");
42     return 0;
43 }
44 //???
45 float suma()
46 {
47     int i, n;
48     float s, x;
49     s=0;
50     printf("Podaj ilosc liczb:");
51     scanf("%d", &n);
52     i=1;
53     while(i<=n) //???
54     {
55         printf("Podaj liczbe:");
56         scanf("%f", &x);
57         if(x > 100) s+=x;
58         i++;
59     }
60     return s;
61 }
62 void tablicaWartosci(float a, float b, float k)
63 {
64     float x;
65     x=a;
66     while(x<=b) //???
67     {
68         printf("x=%f      x+sin(x)=%f\n", x, x+sin(x));
69         x+=k;
70     }
71 }
72 int iloczyn(int a, int b)
73 {
74     int w, il;
75     if (a > b)
76     {
77         w=a;
78         a=b;
79         b=w;
80     }
81 }
```



```
73  if ((a+1) % 3 ==0) a++;
74  else if ( (a+2) % 3 ==0) a+=2;
75  w=a;
76  il=1;
77  while( w <= b) //???
78  {      il*=w;
79          w+=3;
80  }
81  return il;
82 }

83  int zagadka (int a)
84  {      int w, il;
85  il=0;
86  do //???
87  {      printf(" Podaj liczbę ");
88          scanf("%d", &w);
89          il++;
90  } while (a!= w && il < 6);
91  if (a != w) il=-1;
92  return il;
93 }
```

Zadania do wykonania

Zadanie 6.2. Średnia liczb

Napisz funkcję `sredniaWhile` obliczającą średnią arytmetyczną **n** liczb z wykorzystaniem instrukcji **while** i funkcję `sredniaDoWhile` obliczającą średnią arytmetyczną **n** liczb z wykorzystaniem instrukcji **do...while**.

Zadanie 6.3. Średnia liczb parzystych i nieparzystych

Napisz i wywołaj funkcję, która pobiera od użytkownika liczby całkowite do momentu wpisania cyfry 0 i wyświetla średnią liczb parzystych i nieparzystych.

Zadanie 6.4. Zliczanie wystąpień litery i liczby cyfr

Napisz i wywołaj funkcję, która dla wprowadzonego przez użytkownika zdania zakończonego kropką obliczy, ile razy wystąpiła w nim podana litera (np. a) oraz ile było cyfr.



Zadanie 6.5. Znajdowanie największej liczby

Napisz i wywołaj funkcję, która pobiera od użytkownika liczbę, ostatnia jest równa -1 oraz zwraca największą z tych liczb.

Zadanie 6.6. Menu programu

Program główny skonstruuj na zasadzie menu, z którego użytkownik wielokrotnie wybiera pozycję do wykonania lub kończy program.

Zadania dodatkowe

Zadanie 6.7. Ciąg rosnący

Napisz i wywołaj funkcję, która pobiera od użytkownika n liczb i sprawdza czy tworzą ciąg rosnący. Wynik funkcji: 1 jeśli tak, 0 jeśli nie.

Zadanie 6.8. Obliczenie stanu konta bankowego

Napisz i wywołaj funkcję, która dla podanych w parametrach: wpłaty oraz oprocentowania oblicza i wyświetla, po jakim czasie kwota wpłaty będzie co najmniej podwojona i ile dokładnie wynosi.

Zadanie 6.9. Sprawdzenie wyników egzaminu

Napisz i wywołaj funkcję, która dla n studentów sprawdzi, czy wszyscy studenci zdali egzamin.

Zadanie 6.10. Hotel bez pokoju nr.13

Hotel posiada n pokoi ($n > 13$), połowa na parterze, połowa na I piętrze (w przypadku gdy n jest nieparzyste na I piętrze jest o 1 pokój więcej). Napisz funkcję, która wyświetli listę pokoi hotelowych od nr 1 do n (bez nr 13). Jeżeli pokój ma nr nieparzysty to jest on jednoosobowy, jeżeli parzysty to dwuosobowy. Lista powinna zawierać: nr pokoju, jedno/dwu osobowy, parter/I piętro. Powinna być wyświetlona równo w kolumnach. Wykorzystaj instrukcję CONTINUE.

Zadanie 6.11.

Napisz i wywołaj funkcję, która wyświetla kody ASCII liter alfabetu w wierszach w układzie:

```
A -> 65    a -> 97  
B -> 66    b -> 98
```



LABORATORIUM 7. INSTRUKCJA ITERACYJNA FOR.

Cel laboratorium:

Zapoznanie z realizacją algorytmów wymagających wielokrotnego powtarzania sekwencji instrukcji z wykorzystaniem pomocy pętli licznikowej. Nabycie praktycznych umiejętności programowania algorytmów iteracyjnych z wykorzystaniem instrukcji FOR.

Zakres tematyczny zajęć:

- instrukcja FOR,
- operator przecinkowy,
- zagnieżdżenie instrukcji iteracyjnych.

Kompendium wiedzy:

Instrukcja iteracyjna (pętli) for służy do powtarzania fragmentu programu dopóki wartość wyrażenia wpisanego w instrukcję pętli jest różna od zera, czyli wyrażenie jest prawdziwe.

Działanie **for**:

- nadanie zmiennej zm wartości wp,
- sprawdzenie wartości wyrażenia,
- jeśli jest prawdziwe: wykonana jest instrukcja, zmiana wartości zm i powrót do sprawdzenia wartości wyrażenia;
- jeśli nie jest prawdziwe to wykonywane są instrukcje poza zasięgiem pętli.

```
for (zm=wp ; wyrażenie ; zmiana wartości zm)
instrukcja; //zasięg pętli
```

```
//użycie instrukcji złożonej w for
for (zm=wp ; wyrażenie ; zmiana wartości zm)
{instrukcja1; //zasięg pętli
 instrukcja2; //zasięg pętli
 ... //zasięg pętli
}
```

```
np. int k;
for( k=1; k<6; k++)
    printf("*\n"); //p1
int i, x=0;
for (i=-5; i<=5; i+=3) //p2
    {x+=i;
     printf( "x=%d\n", x);}
int a; char ch;
for(a=1,ch='a';ch<'k';a++,ch++)//p3
{printf("%3d. Kod ASCII %c = %d\n",a,ch,ch);}
```



Pytania kontrolne:

1. Podaj składnię instrukcji FOR.
2. Jak interpretowana jest pętla FOR z trzema pustymi wyrażeniami for(; ;)?
3. Do czego służy operator przecinkowy w instrukcji FOR?
4. Objaśnij, na czym polega zagnieżdżanie instrukcji iteracyjnych.

Zadania do analizy

Zadanie 7.1. Instrukcja iteracyjna FOR

- Przeanalizuj przykład programu wykorzystującego instrukcję iteracyjną FOR.

Funkcja **iloczyn** oblicza iloczyn N liczb rzeczywistych.

Funkcja **srednia** oblicza średnią arytmetyczną dodatnich liczb z N podanych liczb.

Funkcja **tablicaWartosci** oblicza i wyświetla wartości wyrażenia $x + \sin x$ dla $x \in \langle a, b \rangle$, z krokiem k.

- Podaj tekst w komentarzach.

```
1 float iloczyn();
2 void srednia();
3 void tablica_wartosci(float, float, float); //???
4
5 int main ( )
6 {   int nr=1;
7     while (nr) //???
8     {   printf ("Wpisz numer funkcji (1, 2 lub 3)\n");
9         printf ("Koniec programu - wpisz 0 ");
10        scanf( "%d", &nr);
11        if (nr==1) printf( "iloczyn liczb = %f",
12                           iloczyn());
13        else
14        if(nr==2) srednia();
15        else
16        if(nr==3)
17        {   float a,b,k;
18            printf(" wpisz konce przedzialu i krok\n");
19            scanf("%f%f%f", &a, &b, &k);
20            tablica_wartosci(a, b, k);
21        }
22        printf("\nKONIEC\n");
23        return 0;
24    }
25 float iloczyn()
26 { int i, n;
27   float s, x, il;
```



```
28  il=1;
29  printf("Podaj ile bedzie liczb:");
30  scanf("%d", &n);
31  for (i=1; i<=n; i++) //???
32  {    printf("Podaj liczbe:");
33      scanf("%f", &x);
34      il*=x;
35  }
36  return il;
37 }
38 void srednia()
39 { int i, n, il;
40   float s, x;
41   s=0;
42   il=0;
43   printf("Podaj ile bedzie liczb:");
44   scanf("%d", &n);
45   for (i=1; i<=n; i++) //???
46   {    printf("Podaj liczbe:");
47       scanf("%f", &x);
48       if (x >0)
49       { il++;
50         s+=x;
51       }
52   }
53   if( il > 0 )
54   {    s=s/il;
55       printf("srednia liczb dodatnich=%f\n", s);
56   }
57   else printf(" Brak liczb dodatnich\n");
58 }
59 void tablica_wartosci(float A, float B, float K)
60 { float x;
61   x=A;
62   for(x=A; x<=B; x+=K) //???
63   {    printf("x=%f      x+sin(x)=%f\n", x, x+sin(x));
64   }
65 }
```

Zadania do wykonania

Zadanie 7.2. Szlaczek

Napisz funkcję wyświetlającą linię podanej długości wybranym znakiem (np. 50 gwiazdek w jednym wierszu). Wywołaj tę funkcję.



Fundusze Europejskie
Wiedza Edukacja Rozwój



**Rzeczpospolita
Polska**

Unia Europejska
Europejski Fundusz Społeczny



Zadanie 7.3. Średnia liczb parzystych

Napisz funkcję obliczającą średnią arytmetyczną liczb parzystych spośród n liczb całkowitych podanych przez użytkownika. Wywołaj tę funkcję.

Zadanie 7.4. Lista Płac

Pracownicy mają otrzymać podwyżkę według następującego algorytmu:

- gdy zarobki > 5000 – podwyżka wynosi 5%,
- gdy zarobki ≤ 5000 – podwyżka wynosi 10%.

Napisz funkcję, która pobiera od użytkownika płace n pracowników, wyświetla nową listę płac oraz sumę nowych płac. Wywołaj tę funkcję.

Zadanie 7.5. Ciąg arytmetyczny

Napisz funkcję sprawdzającą, czy N liczb podanych przez użytkownika tworzy ciąg arytmetyczny. Wywołaj tę funkcję

Wskazówka: liczby tworzą ciąg arytmetyczny, jeśli różnica pomiędzy sąsiednimi liczbami jest stała.

Zadanie 7.6. Suma szeregu

$$\sin x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots$$

Napisz funkcję, która oblicza wartość $\sin x$ za pomocą powyższego wzoru dla K składników sumy. Wywołaj tę funkcję. Pobranie wartości kąta, zmiennej K i wyświetlenie wyniku jest w funkcji `main()`.

Zadania dodatkowe

Zadanie 7.7. Silnia n liczb nieujemnych

Napisz i wywołaj funkcje, które obliczają liczbę $n!$ (n mniejsze niż 13). Jedna z nich wykorzystuje rekurencję, druga nie. Sprawdź ich wynik dla tych samych wartości parametrów.

Zadanie 7.8. Liczby trzycyfrowe

Napisz i wywołaj funkcję wyznaczania wszystkich liczb trzycyfrowych, które są równe sumie sześcianów swoich cyfr, np. $153 = 1^3 + 5^3 + 3^3$.

Zadanie 7.9. Liczby pierwsze

Napisz program, którego zadaniem będzie odnalezienie i wypisanie n kolejnych liczb pierwszych, oraz podanie ile było liczb, które okazały się nie być pierwszymi. Liczba n musi być z przedziału od 1 do 20.



- Odnalezione liczby pierwsze należy przedstawić w kolejności od pierwszej do ostatniej znalezionej.
- Odnalezione liczby pierwsze przedstaw w kolejności odwrotnej.

Zadanie 7.10. Trójki pitagorejskie

Napisz program, którego zadaniem będzie odnalezienie n trójek pitagorejskich (tj. trójki liczb całkowitych a, b, c takich, że $a^2 + b^2 = c^2$), składających się z liczb mniejszych od n . Liczba n musi być z przedziału od 10 do 200.

- Wyniki przedstaw w kolejności od pierwszej znalezionej trójki do ostatniej.
- Dodatkowo, wyniki przedstaw w kolejności odwrotnej.

Zadanie 7.11. Liczby

Napisz funkcje wyznaczania liczb:

- Automorficznych (liczby, które występują na końcu swego kwadratu).
- Pierwszych (liczby naturalne > 1 , które dzielą się tylko przez 1 i samą siebie).
- Pitagorejskich (liczby spełniające warunek: $a^2 + b^2 = c^2$) dla danego przedziału $<m, n>$.

Funkcja main powinna zawierać menu, pozwalające na wybór rodzaju poszukiwanych liczb.



LABORATORIUM 8. KOŁOKWIUM 1

Cel laboratorium:

Weryfikacja nabytych umiejętności pisania prostych programów z wykorzystaniem podstawowych instrukcji sterujących i zmiennych prostego typu.

Wytyczne do kolokwium 1:

- zakres laboratoriów 1-7,
- próg zaliczeniowy 60%,
- pozostałe wytyczne i sposób zaliczenia kolokwium ustala prowadzący zajęcia.



Fundusze Europejskie
Wiedza Edukacja Rozwój



**Rzeczpospolita
Polska**

Unia Europejska
Europejski Fundusz Społeczny



LABORATORIUM 9. FUNKCJE Z ARGUMENTAMI WSKAŹNIKOWYMI.

Cel laboratorium:

Zapoznanie z pojęciem wskaźnika. Nabycie praktycznych umiejętności zastosowania wskaźników do komunikacji między funkcjami.

Zakres tematyczny zajęć:

- ✓ *pojęcie i definicja wskaźnika,*
- ✓ *operatory związane ze wskaźnikami,*
- ✓ *parametry wskaźnikowe w funkcjach.*

Kompendium wiedzy:

Wskaźnik (zmienna wskaźnikowa) jest zmienną, której wartością jest adres innej zmiennej.

Deklaracja wskaźnika:

```
typ *nazwa;
```

Typ powinien być taki, jak typ zmiennej, której adres będzie przechowywany w zmiennej wskaźnikowej.

```
np.  int *wx;
      float *wy;
      char *wz;
```

Operatory związane ze wskaźnikami:

- ***Operator adresu &*** - pozwala uzyskać adres zmiennej, która po nim następuje.
- ***Operator dereferencji (pośredniości) **** – zwraca wartość przechowywaną pod adresem wskazywanym przez zmienną wskaźnikową.

Wskaźnik należy powiązać z zmienną wskazywaną operatorem adresu &:

```
np.  int x;
      float y;
      char z;

      wx = &x;
      wy = &y;
      wz = &z;

      x = 7;
      printf ("adres zmiennej x:%p\n", wx);
      printf ("wartość zmiennej x:%d\n", x);

      *wx = 5; //x = 5;
      printf ("wartość pod adresem wx:%d\n", *wx);
      printf ("wartość zmiennej x:%d\n", x);
```



Wykorzystanie wskaźników do przekazywania parametrów:

```
//prototypy funkcji
int funkcja1(int x); // parametr przekazywany przez wartość
int funkcja2(int *x); //parametr przekazywany przez adres
// wywołanie funkcji
int a=1,b,c;
b=funkcja1(a);
c=funkcja2(&a);
```

Pytania kontrolne:

1. Co to jest wskaźnik w programowaniu? Jak go zadeklarować?
2. Podaj operatory związane ze wskaźnikami i wyjaśnij ich działanie.
3. Jaką rolę pełnią parametry wskaźnikowe stosowane w funkcjach?

Zadania do analizy

Zadanie 9.1. Wskaźniki

- Przeanalizuj przykład programu wykorzystującego wskaźniki:
Funkcja `pr` oblicza pole powierzchni i objętość prostopadłościanu.
Funkcja `w` pole powierzchni i objętość walca.
- Podaj tekst w komentarzach.

```
1 float pr(float, float, float, float *); //???
2 float w(float, float, float *); //???
3 int main ( )
5 { float x, y, pole, obj;
6   int nr;
7   printf ( "program oblicza pole i objętość brył:\n");
8   printf ("prostopadloscian - wpisz 1 \n");
9   printf ("walec - wpisz 2 \n");
10  printf ("Wpisz numer funkcji 1 lub 2\n");
11  scanf( "%d", &nr);
12  if (nr==1)
13  { float z;
14    printf( "podaj długości boków prostopadloscianu ");
15    scanf ("%f%f%f", &x, &y, &z);
16    pole = pr(x, y, z, &obj); //???
17    if(pole != 0)
18      printf (" pole= %f\toobjetosc=%f\n", pole, obj);
19    else
20      printf (" niepoprawne dane\n");
21  }
22  else
23    if (nr==2)
```



```
24     { printf( "podaj promien i wysokosc walca ");
25         scanf ("%f%f", &x, &y);
26         pole = w(x, y, &obj); //???
27         if(pole != 0)
28             printf (" pole= %f\tobjetosc=%f\n", pole, obj);
29         else
30             printf (" niepoprawne dane\n");
31     }
32     else
33         printf (" niepoprawny numer opcji dane\n");
34     return 0;
35 }
36 float pr(float a, float b, float c, float *V) //???
37 { *V = 0 ; // ta instrukcja jest opcjonalna
38     if ( a <= 0 || b <= 0 || c <= 0) return 0;
39     * V= a * b * c; //???
40     return 2 * (a * b + a * c + b * c); //???
41 }
42 float w(float r, float h, float *V) //???
43 { *V = 0 ; // ta instrukcja jest opcjonalna
44     if ( r <= 0 || h <= 0) return 0;
45     * V= M_PI * r * r * h; //???
46     return M_PI * r * (2 * h + r); //???
47 }
```

Zadania do wykonania

Zadanie 9.2. Zamiana miejscami dwóch liczb

Napisz funkcję z dwoma parametrami, która zamienia miejscami dwie liczby. Wywołaj tę funkcję.

Zadanie 9.3. Zamiana wartości dwóch liczb

Napisz funkcję, która pobiera dwie zmienne typu całkowitego i przypisuje im odpowiednio wartość ich sumy i różnicy.

Zadanie 9.4. Iloczyn liczb

Napisz funkcję obliczającą iloczyn liczb z przedziału <1,100> spośród n liczb podanych przez użytkownika. Funkcja zwraca iloczyn poprzez parametr wskaźnikowy a informację, czy były liczby spełniające warunek poprzez return. Wywołaj tę funkcję, wyświetl wynik: wartość iloczynu lub komunikat o braku liczb.



Zadanie 9.5. Część całkowita i ułamkowa liczby rzeczywistej

Napisz funkcję dekomponującą dowolną liczbę rzeczywistą na część całkowitą i rzeczywistą. Wywołaj tę funkcję w main() dla trzech podanych przez użytkownika liczb rzeczywistych.

Zadanie 9.6. Liczby

Napisz funkcję, która pobiera od użytkownika n liczb, zwraca informację ile jest liczb dodatnich i ile jest liczb = 0.

Zadania dodatkowe

Zadanie 9.7. Dwie największe liczby

Napisz funkcję zwracającą 2 największe liczby spośród n różnych liczb podanych przez użytkownika. Wywołaj tę funkcję, wyświetl wyniki.

Zadanie 9.8. Lokaty bankowe

Bank oferuje lokaty: półroczną oprocentowaną p1% (w skali roku) i roczną oprocentowaną p2%, przy czym $p1 < p2$. Klient wpłaca x zł na rok. Napisz funkcję zwracającą w parametrach wskaźnikowych uzyskane po roku wypłaty dla obu wariantów lokat. Wczytanie danych zrealizuj w main(). Wywołaj tę funkcję, wyświetl wyniki.

Zadanie 9.9. Układ równań liniowych

Napisz funkcję rozwiązującą układ 2 równań liniowych metodą wyznaczników. Wynik przekazany przez return:

- 1 - układ ma rozwiązanie
- 0 - układ jest nieoznaczony
- 1 – układ jest sprzeczny.

Rozwiązanie układu czyli wartości x, y przekazane przez parametry wskaźnikowe. Wprowadzenie danych w main(). Wywołaj tę funkcję, wyświetl wyniki.
Wskazówka - Metoda wyznaczników dla układu równań liniowych.

Układ równań:

$$A1 \cdot x + B1 \cdot y = C1$$

$$A2 \cdot x + B2 \cdot y = C2$$

Wyznaczniki:

$$W = A1 \cdot B2 - A2 \cdot B1$$

$$W_x = C1 \cdot B2 - C2 \cdot B1$$

$$W_y = A1 \cdot C2 - A2 \cdot C1$$

Interpretacja wartości wyznaczników:

Jeśli $W = 0$ i $W_x = 0$ i $W_y = 0$ to układ jest nieoznaczony.

Jeśli $W = 0$ i ($W_x \neq 0$ lub $W_y \neq 0$) to układ jest sprzeczny.

Jeśli $W \neq 0$ to $x = W_x / W$, $y = W_y / W$

Zadanie 9.10. Zamówienie

Dane: cena hurtowa 1 sztuki towaru, liczba zamówionych sztuk. Jeśli klient zamawia mniej niż 10 sztuk, to płaci cenę detaliczną, która jest o 20% wyższa. Napisz funkcję, która zwraca koszt zamówienia i cenę 1 sztuki towaru w tym zamówieniu.

Zadanie 9.11. Położenie punktu

Napisz funkcję, która zwraca odległość punktu od początku układu współrzędnych oraz informację o położeniu tego punktu (numer ćwiartki układu współrzędnych).

LABORATORIUM 10. ZŁOŻONE TYPY DANYCH. TABLICE STATYCZNE JEDNO I WIELOWYMIAROWE.

Cel laboratorium:

Zapoznanie z typem złożonym – tablicą. Nabycie praktycznych umiejętności zastosowania tablic w programie.

Zakres tematyczny zajęć:

- *pojęcie tablicy,*
- *rodzaje tablic,*
- *sposoby deklaracji tablic,*
- *typowe operacje na tablicach,*
- *tablice jako parametry funkcji.*

Kompendium wiedzy:

Tablica:

- jest uporządkowanym zbiorem zmiennych **tego samego typu**,
 - jest poindeksowaną kolekcją elementów tego samego typu.
- Kolejność elementów w zbiorze określona jest poprzez indeks/indeksy.

Deklaracja tablicy jednowymiarowej i dwuwymiarowej:

```
typ nazwa[liczba elementów]; //jedno  
typ nazwa[liczba wierszy][liczba kolumn]; //dwu
```

Tablice:

- o stałej długości:
`#define N 5 int tab1[N];`
- o zmiennej długości:
`int n; scanf("%d",&n); int tab2[n];`

Zmienna indeksowa musi być typu całkowitego i przyjmuje wartości nieujemne. Pierwszy element tablicy ma indeks równy 0.

Zapis:

`tab[0]` oznacza pierwszy element tablicy jednowymiarowej

`tab2[0][0]` oznacza pierwszy element tablicy dwuwymiarowej

Nazwa tablicy jest jednocześnie wskaźnikiem na pierwszy element.

```
int tab[5]={1,2,3,4,5};  
//inicjacja tablicy podczas deklaracji  
tab       <->   &tab[0] ;  
tab+2     <->   &tab[2] ;   //ten sam adres  
*(tab+2) <->   tab[2] ;   //ta sama wartość
```

`tab[i]==*(tab+i)`



Praca na tablicy

Przykład 1

```
int tab[10], i;  
for (i=0; i<10; i++)  
tab[i]= i * i; // zapis indeksowy
```

Przykład 2

```
int tab[10], i;  
for (i=0; i<10; i++)  
*(tab + i)= i *i; // zapis wskaźnikowy
```

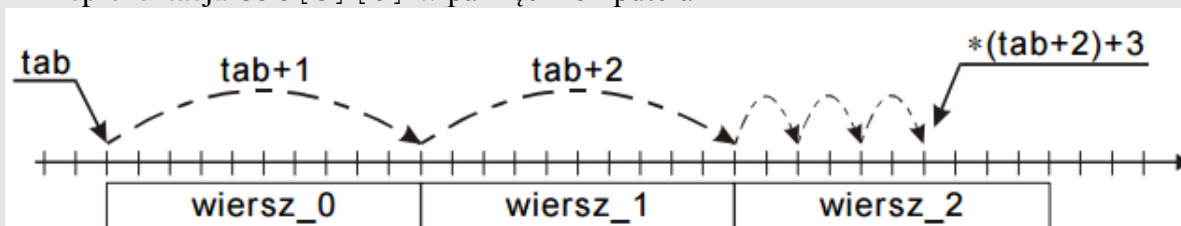
Tablice dwuwymiarowe

```
int tab[3][5];  
int i, j ;
```

$$\text{tab}[i][j] == *(*(\text{tab} + i) + j)$$

np. `tab[0][0] == *(*(tab+0)+0) == **tab`

Reprezentacja `tab[3][5]` w pamięci komputera



Pytania kontrolne:

1. Co to jest tablica? Objasnij pojęcie zmienna tablicowa i zmienna indeksowana.
2. Jakie są rodzaje tablic?
3. Jak zadeklarować tablicę? Jak określić liczbę jej elementów?
4. W jaki sposób zdefiniować tablicę z konkretnymi danymi?
5. Jak wczytać dane do tablicy i jak je wyświetlić?
6. Jak odwołać się do elementu tablicy za pomocą notacji tablicowej, a jak za pomocą notacji wskaźnikowej?
7. Jak zdefiniować tablicę jako parametr funkcji?

Zadania do analizy

Zadanie 10.1. Tablice statyczne

- Przeanalizuj przykład programu wykorzystującego tablice:
Funkcja `czytaj` wczytuje dane do tablicy z klawiatury.
Funkcja `roznica` wyświetla różnicę między największym i najmniejszym elementem.
Funkcja `nowa` przepisuje liczby dodatnie do nowej tablicy.



Fundusze Europejskie
Wiedza Edukacja Rozwój



Rzeczpospolita
Polska

Unia Europejska
Europejski Fundusz Społeczny



- Podaj tekst w komentarzach

```
1 void czytaj(int n, float a[]); //???
2 float roznica(int n, float T[]); //???
3 int nowa(int n, float A[], float B[]); //???
4 int main ( )
5 { int i, n, k;
6   printf( "Program wczytuje liczby do tablicy,\n");
7   printf( "wyswietla roznice pomiedzy najwiekszym");
8   printf( " i najmniejszym elementem tablicy,\n");
9   printf( "przepisuje liczby dodatnie do nowej
      tablicy\n");
10  printf( "oraz wyswietla te tablice\n\n");
11  printf("Wpisz liczbe elementow tablicy: ");
12  scanf("%d", &n);
13  float T[n]; //???
14  float NT[n]; //???
15  czytaj(n, T); //???
16  printf ("max-min=%f \n",roznica(n, T)); //???
17  k = nowa(n, T, NT); //???
18  printf ("\nNowa tablica \n");
19  for (i=0; i<k; i++)
20  printf ( "%.2f\n", NT[i]); //???
21  return 0;
22 }

23 void czytaj(int n, float a[]) //???
24 { int i;
25   for (i=0; i<n; i++)
26   { printf("wpisz liczbe a[%d]: ", i);
27     scanf("%f", &a[i]); //???
28   }
29 }
30 float roznica(int n, float T[]) //???
31 { int i;
32   float max, min=T[0];
33   max=min;
34   for (i=1; i<n; i++)
35   { if (T[i] > max ) max=T[i];
36     else
37       if (T[i] < min ) min=T[i];
38   }
39   return max-min;
40 }
41 int nowa(int n, float A[], float B[]) //???
42 { int i, j=0;
43   for (i=0; i<n; i++)
44   if (A[i] > 0)
45   { B[j] = A[i];
```



```
46         j++;  
47     }  
48     return j;  
49 }
```

Zadania do wykonania

Zadanie 10.2. Typowe operacje na tablicy jednowymiarowej liczb całkowitych z zastosowaniem notacji tablicowej

Napisz, wykorzystując notację tablicową, funkcje: `wczytaj`, `wyswietl`, `max`, `suma`, `ileRazy`, które dla n elementowej jednowymiarowej tablicy liczb całkowitych pozwolą na:

- wczytanie danych do tablicy z klawiatury,
- wyświetlenie elementów tablicy na ekranie,
- obliczenie największej wartości,
- obliczenie sumy elementów tablicy,
- zliczenie, ile razy wystąpiła w tablicy podana przez użytkownika liczba.

W funkcji `main` zadeklaruj tablicę `tab1` o stałej długości N (stała) i tablicę `tab2` o zmiennej długości n (zmienna). Wczytaj i wyświetl elementy tych tablic, oblicz `max` i sumy ich elementów. Sprawdź, ile razy wystąpiła w nich podana liczba.

Zadanie 10.3. Typowe operacje na tablicy jednowymiarowej liczb całkowitych z zastosowaniem notacji wskaźnikowej

Napisz, z wykorzystaniem notacji wskaźnikowej, funkcje: `min`, `srednia`, `losuj`, `ileRazy2`, które dla n elementowej jednowymiarowej tablicy liczb całkowitych pozwolą na:

- obliczenie najmniejszej wartości,
- obliczenie średniej elementów tablicy,
- wypełnienie tablicy wylosowanymi liczbami jednocyfrowymi
- zliczenie, ile razy wystąpiła w tablicy podana przez użytkownika liczba.

W funkcji `main` zdefiniuj tablicę `tab3` liczb całkowitych z zainicjowanymi wartościami (z podaniem jej rozmiaru) i tablicę `tab4` liczb całkowitych z zainicjowanymi wartościami (bez podania jej rozmiaru) i n elementową tablicę `tab5`, którą należy wypełnić wylosowanymi jednocyfrowymi liczbami i oblicz dla tych tablic `min` i `średnią`. Sprawdź, ile razy wystąpiła w nich podana liczba.

Zadanie 10.3. Operacje na dwuwymiarowej tablicy liczb rzeczywistych

Napisz, z wykorzystaniem notacji tablicowej, funkcje, które dla tablicy liczb rzeczywistych o wymiarach N wierszy i M kolumn pozwolą na:

- wczytanie danych z klawiatury do tablicy,
- wyświetlenie elementów tablicy w postaci tabeli (w równych kolumnach),
- obliczenie sumy i średniej jej elementów.



Przetestuj działania tych funkcji w programie z menu.

Zadanie 10.4. Przekątne tablicy

Napisz funkcje, które dla dwuwymiarowej kwadratowej tablicy liczb rzeczywistych o rozmiarze $N \times N$ pozwolą na:

- obliczenie iloczynu elementów na głównej przekątnej,
- obliczenie sumy elementów nad główną przekątną.

Napisz program, w którym wczytane są dane, wywołane funkcje, wyświetlone wyniki.

Zadanie 10.5. Kopiowanie tablic

Napisz program, który inicjalizuje tablicę, a następnie kopiuje jej zawartość do dwóch innych tablic (wszystkie trzy tablice powinny być zadeklarowane w funkcji `main`). Do wykonania pierwszej kopii użyj funkcji wykorzystującej notację tablicową. Do wykonania drugiej kopii użyj funkcji wykorzystującej zapis wskaźnikowy i zwiększanie wskaźników. Każda funkcja powinna przyjmować jako argumenty nazwę tablicy źródłowej, nazwę tablicy docelowej oraz rozmiar tablic.

Zadanie 10.6. Sprawdzanie wartości w tablicy

Napisz funkcję typu logicznego, która dla tablicy z ocenami egzaminu n studentów sprawdzi, czy wszyscy studenci zdali. Przetestuj funkcję w `main()`.

Typ logiczny w C można wykorzystywać za pomocą:

- Wartości wyrażenia: fałsz to wartość 0, prawda wartość różna od zera,
- Dyrektyw preprocesora: `#define FALSE 0 ... #define TRUE !(FALSE)`

Zadania dodatkowe

Zadanie 10.7. Funkcje modyfikujące i tworzące jednowymiarową tablicę liczb

Dane: tablica N różnych liczb rzeczywistych.

Napisz funkcje realizujące następujące zadania:

- wyświetlenie informacji o dostępnych funkcjach (menu) oraz pobranie od użytkownika numeru wybranej funkcji przekazanie go do `main()`,
- zamiana miejscami elementu maksymalnego i minimalnego,
- zapisanie elementów tablicy w odwrotnej kolejności,
- utworzenie nowej tablicy Y zawierającej kwadraty danych liczb, oraz nowej tablicy Z , zawierającej sześciany danych liczb.

Napisz program, w którym wczytane są dane, wywołana funkcja menu, a następnie funkcja wybrana przez użytkownika. Wyświetl wyniki działania funkcji.

Zadanie 10.8. Tworzenie tablic

Dane: tablica liczb rzeczywistych o wymiarach N wierszy, K kolumn.

Napisz funkcje:



- F1 - tworzy tablicę D zawierającą elementy dodatnie,
- F2 – tworzy tablicę U zawierającą elementy ujemne,
- F3 - oblicza ile elementów jest = 0.

Napisz program, w którym wczytane są dane, wywołana funkcja, wyświetlone wyniki.

Zadanie 10.9. Funkcje modyfikujące dwuwymiarową tablicę liczb

Dane: tablica liczb rzeczywistych o wymiarach N wierszy, M kolumn.

Napisz funkcję, która modyfikuje tablicę w następujący sposób: jeśli ponad połowa elementów w kolumnie jest=0, to należy wyzerować pozostałe elementy w tej kolumnie (powtórzyć dla wszystkich kolumn). Utworzyć tablicę B zawierającą numery zmienionych kolumn.

Napisz program, w którym wczytane są dane, wywołana funkcja, wyświetlone wyniki.

Zadanie 10.10. Oceny studentów

Napisz program, który prosi użytkownika o podanie ocen N zespołów studentów, każdy zespół liczy M studentów, a następnie:

- zapisuje te dane w tablicy o wymiarach N x M,
- oblicza średnią dla każdego zespołu,
- oblicza średnią ocen dla wszystkich studentów,
- znajduje najlepszą ocenę spośród wszystkich studentów,
- wyświetla wyniki.

Każde podstawowe zadanie powinno być realizowane przez osobną funkcję.

Zadanie 10.11. Badanie symetryczności tablicy dwuwymiarowej

Dane: tablica liczb rzeczywistych o wymiarach N wierszy, K kolumn.

Napisz funkcję, która pobiera od użytkownika dwuwymiarową tablicę liczbową. Jeśli ilość kolumn jest nieparzysta, to funkcja sprawdza, czy tablica jest symetryczna względem środkowej kolumny, Jeśli ilość kolumn jest parzysta, to funkcja sprawdza, czy brzegowe wiersze są takie same. Funkcja wyświetla wynik. Wywołaj tę funkcję.





Materiały zostały opracowane w ramach projektu
„Zintegrowany Program Rozwoju Politechniki Lubelskiej – część druga”,
umowa nr **POWR.03.05.00-00-Z060/18-00**
w ramach Programu Operacyjnego Wiedza Edukacja Rozwój 2014-2020
współfinansowanego ze środków Europejskiego Funduszu Społecznego