INFORMATYKA

C

WYDZIAŁ ELEKTROTECHNIKI I INFORMATYKI
ELEKTROTECHNIKA
DR INŻ. MICHAŁ ŁANCZONT

IV

Zakres wykładu

1. *Programowanie strukturalne i proceduralne,*
2. *Funkcje,*
3. *Prototypy funkcji,*
4. *Pliki nagłówkowe,*
5. *Wskaźniki*

Programowanie strukturalne

Paradygmat programowania opierający się na podziale kodu źródłowego programu na procedury i hierarchicznie ułożone bloki z wykorzystaniem struktur kontrolnych w postaci instrukcji **wyboru** i **pętli**.

Programowanie proceduralne

Paradygmat programowania zalecający dzielenie kodu na procedury, czyli fragmenty wykonujące ściśle określone operacje.

Procedury nie powinny korzystać ze zmiennych globalnych (*w miarę możliwości*), lecz pobierać i przekazywać wszystkie dane (*czy też wskaźniki do nich*) jako parametry wywołania.

Programowanie proceduralne

- W języku C w kodzie programu misi występować przynajmniej jedna funkcja **main()**
- W kodzie pisanego programu można wydzielić określone zadania z funkcji **main()** do funkcji zadaniowych
- W niektórych językach programowania istnieje rozróżnienie na procedury i funkcje

Programowanie proceduralne

- W klasycznym rozumieniu funkcja jest to wydzielony fragment kodu o określonej nazwie który wykonuje zakodowane w niej zadanie zwracając określoną wartość.

- Procedura jest funkcją która nie zwraca żadnej wartości. W języku C pod tym pojęciem rozumiemy funkcje typu **void**.

Funkcje

- W języku C do kodu pisanego programu można podpiąć bibliotekę funkcyjną
- W języku C zewnętrznych bibliotek za pomocą podpina się do kodu za pomocą dyrektywy

```
#include <nazwa.h>
```

- Biblioteki podzielone są „tematycznie” – operacje wejścia-wyjścia, matematyczne, liczby zespolone, czas itd..

Funkcje

Deklaracja funkcji:

1. Typie zwracanej wartości (funkcja może nie zwracać żadnej wartości – typ **void**)
2. Nazwie funkcji – prawidłowy dobór nazwy funkcji ma decydujące znaczenie dla czytelności kodu programu
3. Liczbie oraz typach parametrów wejściowych – funkcja może nie mieć żadnych parametrów wejściowych

Funkcje

```
typ_danej_1 nazwa(typ_danej_2 zmienna);
```

- typ_danej_1 – określa rodzaj wartości zwracanej przez funkcję. Funkcja może zwracać wartości dowolnego typu możliwego do zdefiniowania w języku C.
- Typ void ma szczególne znaczenie – oznacza że funkcja nie zwraca żadnej wartości.
- Tego typu funkcje w niektórych językach programowania określa się mianem procedury

Funkcje

```
typ_danej_1 nazwa(typ_danej_2 zmienna);
```

- **nazwa** – deklaruje się nazwę tworzonej funkcji, należy pamiętać o katalogu nazw zastrzeżonych.

- Funkcja główna **main()** jest uruchamiana standardowo jako pierwsza.

- W nawiasie deklaruje się parametry wejściowe funkcji, jest to lista zmiennych lokalnych.

Funkcje

```
typ_danej_1 nazwa(typ_danej_2 zmienna);
```

- Funkcja może nie posiadać żadnych parametrów wejściowych jednak wtedy zalecane jest aby w nawiasie deklaracji parametrów umieścić słowo kluczowe `void`

- Wywołanie funkcji sprowadza się do podania jej nazwy wraz z umieszczonymi w nawiasie parametrami wywołania.

```
zmienna_1=nazwa(zmienna_2);
```

Przykład

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int A,B,C;
    printf("Program liczy sume liczba A i B\n");
    printf("Podaj liczbe A=");
    scanf("%i",&A);
    printf("Podaj liczbe B=");
    scanf("%i",&B);
    C=A+B;
    printf("Suma liczba %i+%i wynosi %i",A,B,C);
    return 0;
}
```

```
C:\Windows\system32\cmd.exe
Program liczy sume liczba A i B
Podaj liczbe A=11
Podaj liczbe B=34
Suma liczba 11+34 wynosi 45
Press any key to continue . . .
```

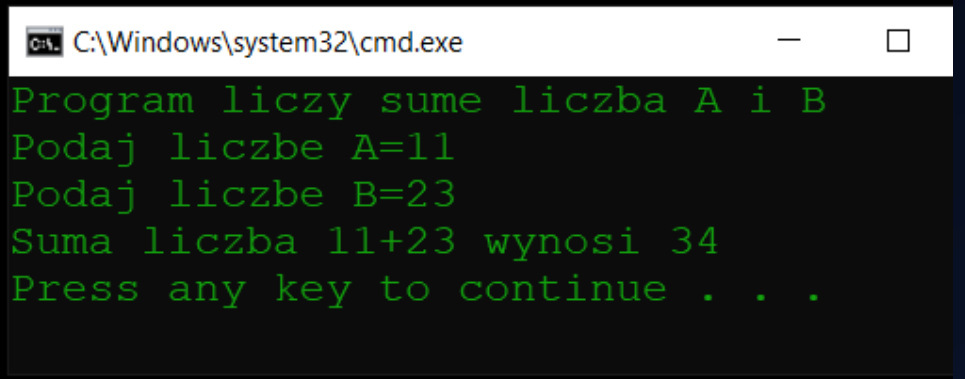
Funkcje

- W chwili wywołania nazwa funkcji musi być znana
  - Funkcja jest dostarczana przez bibliotekę podpiętą dyrektywą #include
  - Definicja funkcji jest w programie umieszczony przed jej wywołanie
  - Funkcja została zadeklarowana

- Kompilator zwróci błąd (nieznana funkcja) jeżeli jeden z podanych warunków nie zostanie wcześniej spełniony

Przykład

```c
#include <stdio.h>
#include <stdlib.h>
int dane(char D)
{
    printf("Podaj liczbe %c=",D);
    int A;
    scanf("%i",&A);
    return A;
}
int main()
{
    printf("Program liczy sume liczba A i B\n");
    int A = dane('A');
    int B = dane('B');
    printf("Suma liczba %i+%i wynosi %i",A,B,A+B);
    return 0;
}
```



Przykład

```c
#include <stdio.h>
#include <stdlib.h>
int dane(char);
int main()
{
    char C='A';
    printf("Program liczy sume liczba A i B\n");
    int A = dane(C);
    int B = dane(C+1);
    printf("Suma liczba %i+%i wynosi %i",A,B,A+B);
    return 0;
}
int dane(char D)
{
    printf("Podaj liczbe %c=",D);
    int A;
    scanf("%i",&A);
    return A;
}
```

```
C:\Windows\system32\cmd.exe
Program liczy sume liczba A i B
Podaj liczbe A=11
Podaj liczbe B=23
Suma liczba 11+23 wynosi 34
Press any key to continue . . .
```

Funkcje

- Przy kodzie zawierającym większą liczbę funkcji zalecane jest ich wcześniejsze deklarowanie
- Kod funkcji może być następnie umieszczony w programie po kodzie funkcji `main()`, np. w kolejności alfabetycznej
- Jest to szczególnie istotne gdy funkcje są zagnieżdżone, tj. jedna funkcja wywołuje inną
- W przypadku zapisu programu bez deklarowania funkcji istotna jest kolejność w jakiej zapisywany jest kod funkcji. W procesie kompilacji może pojawić się błąd lub ostrzeżenie.

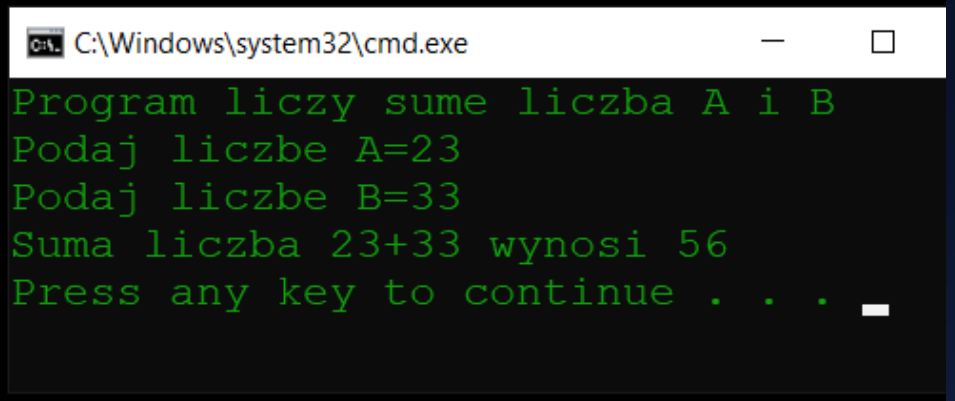
Przykład

```c
#include <stdio.h>
#include <stdlib.h>
int dane(char D)
{
    printf("Podaj liczbe %c=",D);
    return wczytaj();
}
int wczytaj()
{
    int A;
    scanf("%i",&A);
    return A;
}
int main()
{
    char C='A';
    printf("Program liczy sume liczba A i B\n");
    int A = dane(C);
    int B = dane(C+1);
    printf("Suma liczba %i+%i wynosi %i",A,B,A+B);
    return 0;
}
```

```
C:\Windows\system32\cmd.exe
Program liczy sume liczba A i B
Podaj liczbe A=11
Podaj liczbe B=23
Suma liczba 11+23 wynosi 34
Press any key to continue . . .
```

Przykład

```c
#include <stdio.h>
#include <stdlib.h>
int dane(char);
int wczytaj();
int main()
{
    char C='A';
    printf("Program liczy sume liczba A i B\n");
    int A = dane(C);
    int B = dane(C+1);
    printf("Suma liczba %i+%i wynosi %i",A,B,A+B);
    return 0;
}
int dane(char D)
{
    printf("Podaj liczbe %c=",D);
    return wczytaj();
}
int wczytaj()
{
    int A;
    scanf("%i",&A);
    return A;
}
```



Podział kodu

- W języku C można kod programu podzielić na kilka części w celu poprawy przejrzystości kodu
- Podział ten jest realizowany za pomocą dyrektywy preprocesora

```
#include "nazwa_pliku"
```

- W trakcie kompilacji pliki łączone są w jeden plik źródłowy
- W podziale zazwyczaj wydziela się pliki nagłówkowe i źródłowe

Plik nagłówkowy

- Plik nagłówkowy (header) z rozszerzeniem *.h jest plikiem deklaracyjnym
  - Deklarację bibliotek (stdlib.h,...)
  - Deklaracje funkcji
  - Deklaracje zmiennych globalnych
  - Deklaracje stałych

- Plik nagłówkowy zazwyczaj ma nazwę taką samą jak plik źródłowy (source *.c) dla którego jest dedykowany

Plik nagłówkowy

- Zawartość pliku nagłówkowego może zostać tylko raz podpięta do projektu na etapie kompilacji

- Konieczne jest zabezpieczenie przed wielokrotnym podpięciem bibliotek i deklaracją funkcji i zmiennych

```
#ifndef FUNKCJE_H
#define FUNKCJE_H
    //kod pliku nagłówkowego
#endif
```

Plik źródłowy

- Plik z rozszerzeniem *.c zawierający kod funkcji zadeklarowanych w odpowiadającym pliku nagłówkowym
- Na początku pliku źródłowego podpina się za pomocą dyrektywy INCLUDE właściwy plik nagłówkowy
- W odróżnieniu od bibliotek kompilatora nazwy plików nagłówkowych należy umieścić w cudzysłowach.

```
#include ”nagłowek.h”
```

Przykład

program.c ● | program.h | funkcje.c ×

funkcje.c > wczytaj()

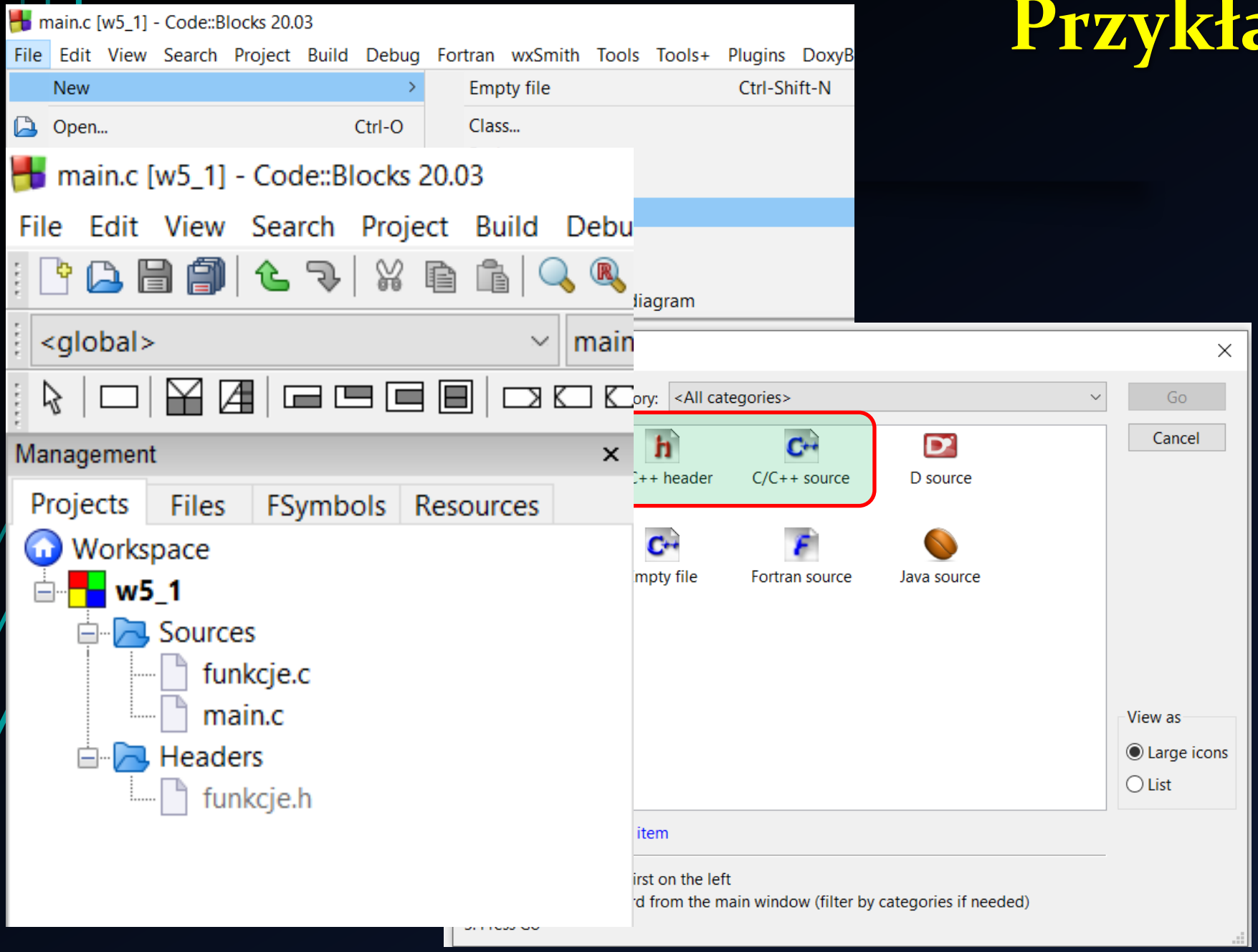
```
1  int dane(char D)
2  {
3      printf("Podaj liczbe %c=",D);
4      return wczytaj();
5  }
6  int wczytaj()
7  {
8      int x;
9      scanf("%i",&x);
10     return x;
11 }
```

```
liczb.\n");

%i\n",A,B,A+B);
```

C:\WINDOWS\system32\cmd.exe

```
Program oblicza sume dwoch liczb.
Podaj liczbe A=23
Podaj liczbe B=56
Suma liczb 23 i 56 wynosi 79

Press any key to continue . . .
```

Przykład


main.c [w5_1] - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyB
New
Empty file Ctrl-Shift-N
Open... Ctrl-O
Class...
main.c [w5_1] - Code::Blocks 20.03
File Edit View Search Project Build Debu
<global>
Management
Projects Files FSymbols Resources
Workspace
w5_1
Sources
funkcje.c
main.c
Headers
funkcje.h
<All categories>
Go
Cancel
C/C++ source
D source
Fortran source
Java source
View as
Large icons
List

Funkcje

iż **void** powinna
erającą polecenie

je przerwanie
zwrócenie przez
przypisana do

```c
#include <stdio.h>

int main()
{
    printf("Test instrukcji return");
}
```

```
E:\Wyk|ady_kod\w5_2\bin\Debug\w5_2.exe
Test instrukcji return
Process returned 0 (0x0)   execution time : 0.032 s
Press any key to continue.
```

- W przy
  funkcja
  zwracar
- W funk
  wielokr
  warunk

```c
#include <stdio.h>

int main()
{
    printf("Test instrukcji return");
    return 5;
}
```

```
E:\Wyk|ady_kod\w5_2\bin\Debug\w5_2.exe
Test instrukcji return
Process returned 5 (0x5)   execution time : 0.030 s
Press any key to continue.
```

Zmienne lokalne i globalne

- W języku C zmienne można zadeklarować w
- nne
- w
- są
- ane
- ntu
- ej)
- W
- ną
- Z
- ną

globalną

```c
#include <stdio.h>
int x=20;
int main()
{
    printf("Zmienna globalna x wynosi: %i\n",x);
    int x=50;
    printf("Zmienna lokalna x wynosi: %i\n",x);
}
```

```
C:\Windows\system32\cmd.exe
Zmienna globalna x wynosi: 20
Zmienna lokalna x wynosi: 50

Press any key to continue . . .
```

Zmienne lokalne

- Zmienne lokalne powinny być deklarowane z przedrostkiem auto

```c
#include <stdio.h>
void drukuj();
int main()
{
    printf("Test zmiennych automatycznej i statycznej.\n");
    drukuj();
    drukuj();
    drukuj();
    drukuj();
}
void drukuj()
{
    auto int x=1;
    static int y=1;
    printf("x=%i\ty=%i\n",x,y);
    x++;
    y++;
}
```

```
C:\Windows\system32\cmd.exe
Test zmiennych automatycznej i statycznej.
x=1     y=1
x=1     y=2
x=1     y=3
x=1     y=4

Press any key to continue . . .
```

static int x=20;

Funkcje

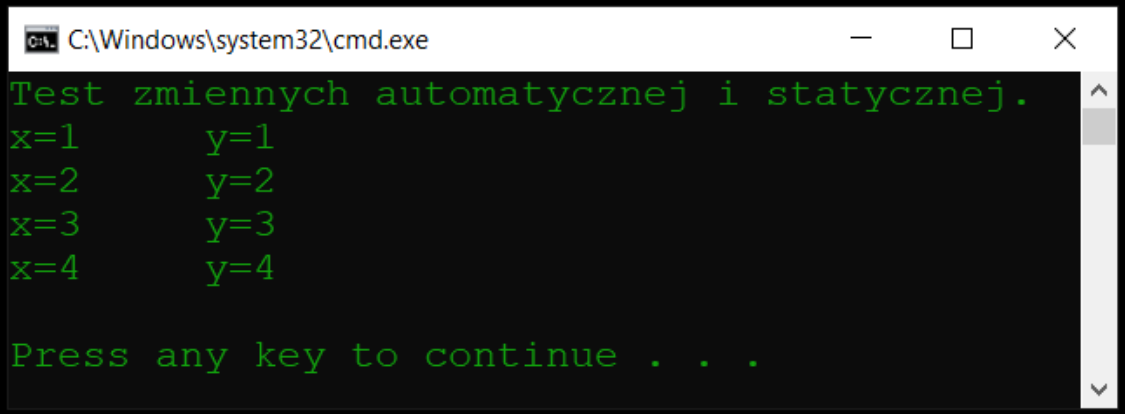
- Funkcja danego typu może zwrócić pojedynczą wartość

```
int licz(int,int);
```

- Funkcja `licz` w wyniku działania zwróci wartość całkowitą
- Są jednak sytuacje kiedy oczekiwane jest aby funkcja w wyniku swojego działania mogła zwracać więcej niż jedną wartość.
- Możliwe jest to przez stosowanie w kodzie programu ze zmiennych globalnych
- Drugą możliwością jest zastosowanie wskaźników

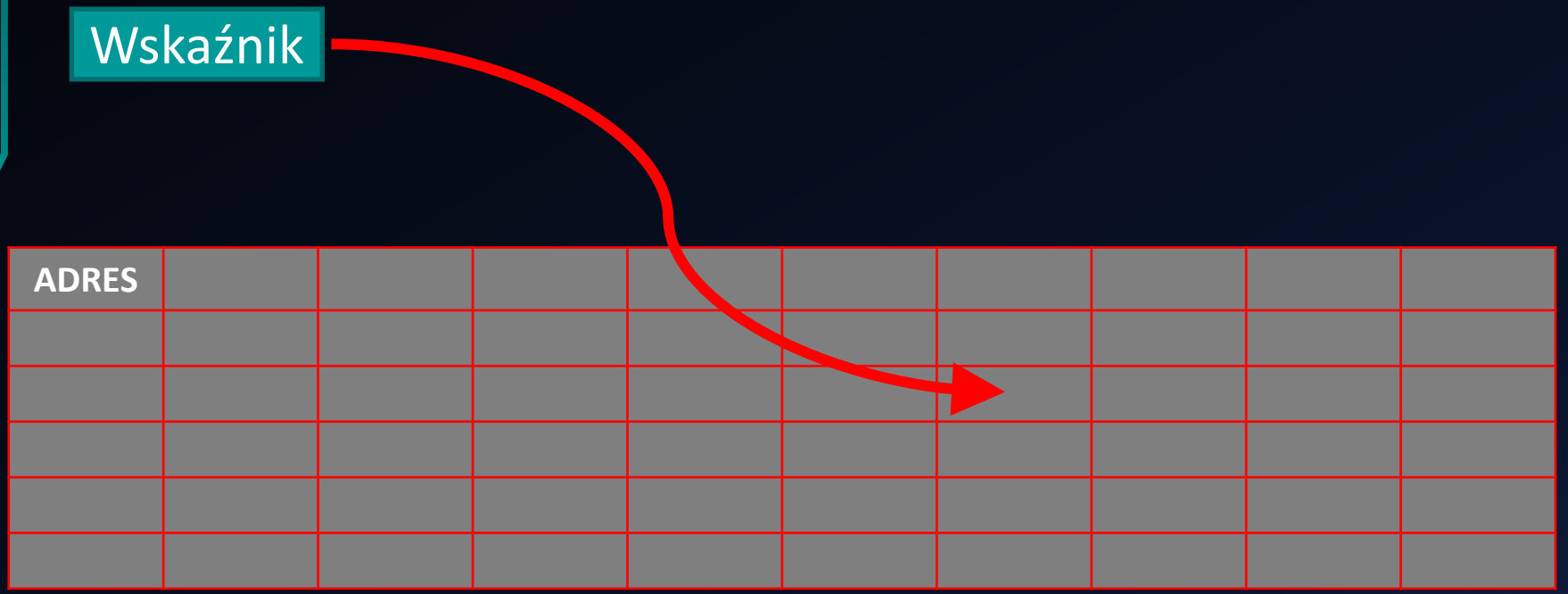
Przykład

```c
#include <stdio.h>
int x=1;
void drukuj();
int main()
{
    printf("Test zmiennych automatycznej i statycznej.\n");
    drukuj();
    drukuj();
    drukuj();
    drukuj();
}
void drukuj()
{
    static int y=1;
    printf("x=%i\ty=%i\n",x,y);
    x++;
    y++;
}
```



Wskaźnik

- Wskaźnik jest zmienną wskazującą na inną zmienną tego samego typu
- Wskaźnik przechowuje adres do zmiennej na którą wskazuje



Wskaźnik

- Wskaźnik deklaruje się podobnie jak klasyczną zmienną

```
typ *nazwa;
```

- Nazwę wskaźnika poprzedza znak * informujący kompilator że dana zmienna będzie wskaźnikiem
- Wskaźnik może wskazywać tylko zmienną tego samego typu co on
- Zadeklarowany wskaźnik, podobnie jak zmienna będzie zawierał losową wartość

Wskaźnik

- Przypisanie adresu do wskaźnika realizuje operator referencji **&**

```
wskaznik = &zmienna;
```

- Odczytanie wartości którą wskaźnik wskazuje realizuje się za pomocą operatora wyłuskania *****

```
printf(„x=%i”,*wskaźnik);
```

Wskaźnik

- Wskaźnik może być także inicjowany jak klasyczna zmienna

```
int *wskaźnik = &zmienna;
```

- Inicjując wskaźnik można skorzystać z deklaratora `auto`

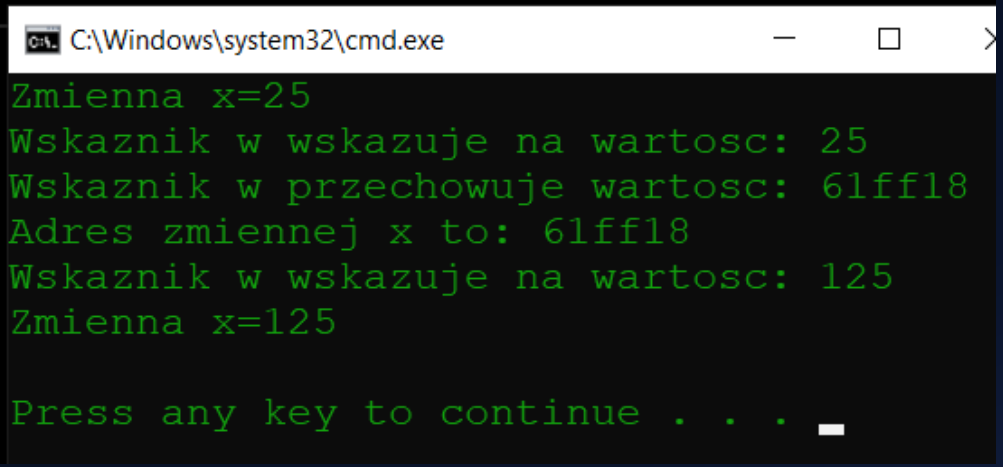
```
auto *wskaźnik = &zmienna;
```

- Za pośrednictwem wskaźnika można zmieniać zawartość komórki pamięci na którą wskazuje
- W ten sposób zmienia się wartość zmiennej na która wskaźnik wskazuje

```
*wskaźnik = 133;
```

Przykład

```c
#include <stdio.h>

int main()
{
    int x = 25;
    printf("Zmienna x=%i\n",x);
    int *w;
    w = &x;
    printf("Wskaznik w wskazuje na wartosc: %i\n",*w);
    printf("Wskaznik w przechowuje wartosc: %x\n",w);
    printf("Adres zmiennej x to: %x\n",&x);
    *w = 125;
    printf("Wskaznik w wskazuje na wartosc: %i\n",*w);
    printf("Zmienna x=%i\n",x);
}
```



Funkcja i zmienna

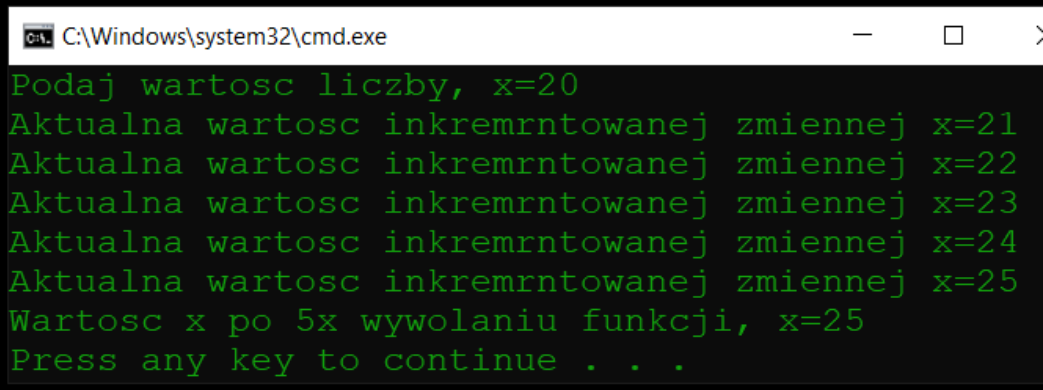
- W klasycznej konstrukcji deklaracji funkcji z parametrami wejściowymi jako zmiennymi, do funkcji przekazywana jest „kopia danych”

```
int funkcja(int x,int y);


zmienna = funkcja(x,y);
```

- Na czas działania funkcji tworzone są zmienne lokalne `x` i `y`, przechowujący w momencie wywołania funkcji wartości liczbowe odpowiednio zmiennych `X` i `Y`
- Zmiany wartości zmiennych `x` i `y` nie wpłyną na wartości zmiennych `X` i `Y`

Przykład

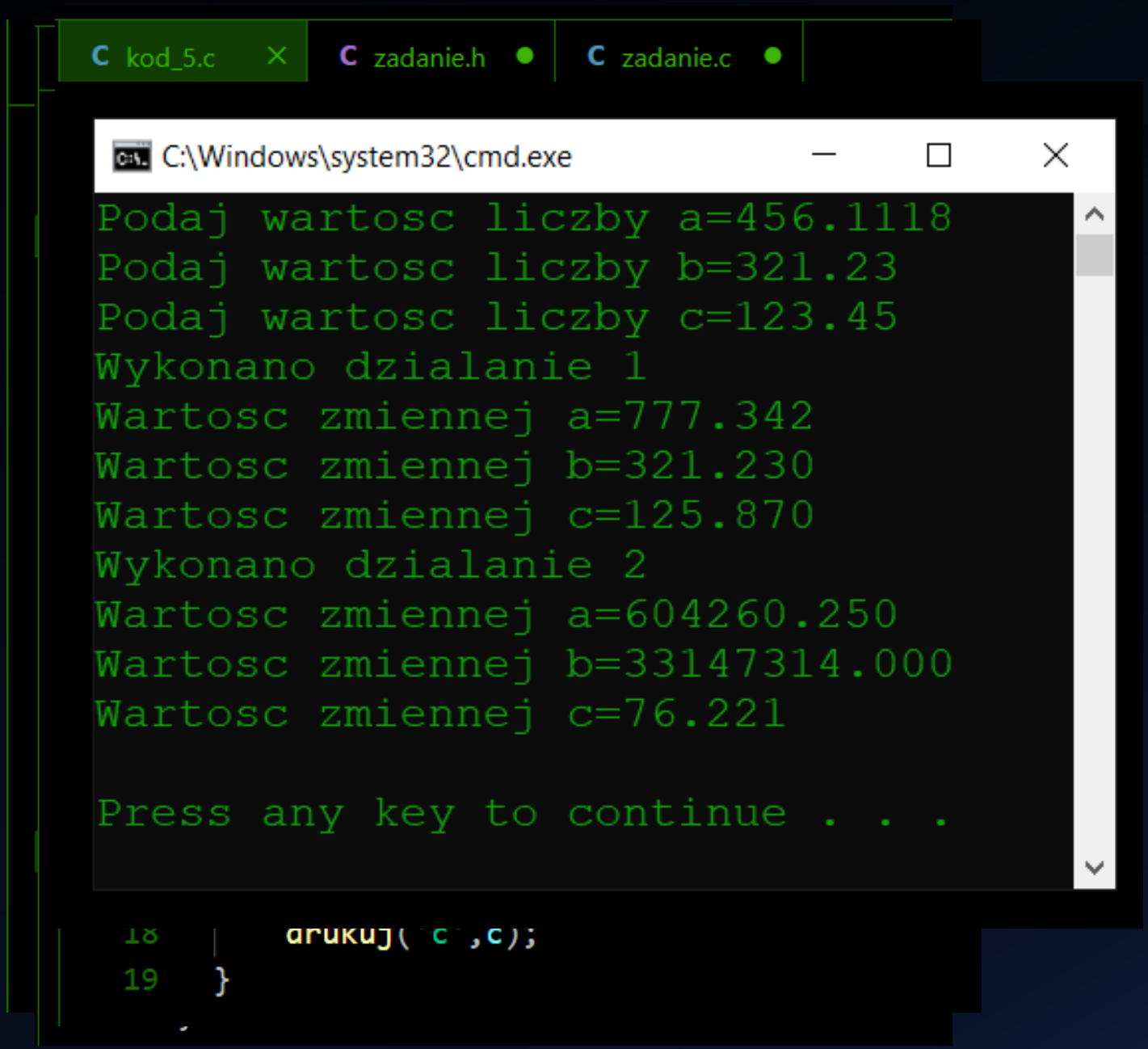
```c
#include <stdio.h>

void inkrementacja(int *x);

int main()
{
    printf("Podaj wartosc liczby, x=");
    int x;
    scanf("%i",&x);
    inkrementacja(&x);
    inkrementacja(&x);
    inkrementacja(&x);
    inkrementacja(&x);
    inkrementacja(&x);
    printf("Wartosc x po 5x wywolaniu funkcji, x=%i",x);
}
void inkrementacja(int *x)
{
    (*x)++;
    printf("Aktualna wartosc inkremrntowanej zmiennej x=%i\n",*x);
}
```



Podsumowanie

- Napisać program wykonujący na podanych dwóch liczbach przykładowe (dowolne) operacje matematyczne
- W programie deklaracje bibliotek, funkcji i zmiennych globalnych wyłączono do pliku nagłówkowego
- Zaprogramować funkcje zwracające 1, 2 i 3 wartości zmiennych
- Program operuje na liczbach rzeczywistych z dokładnością do 3 miejsc po przecinku

Podsumowanie



Biblioteki

- P… …ość d… za p… …owo p…
- P… …ocą d… …ekę u… …ji i ty…
- In… …ości zr… …C w In…

- reference - C++ Reference (cplusplus.com)
- standard C • C++ « PDC (cppox.pl)

stdio.h

- Standardowa biblioteka wejścia-wyjścia
- Dostarcza funkcji umożlwiających komunikację użytkownika z programem (ekran, klawiatura),
- Obsługę błędów na standardowym wejściu-wyjściu
- Zapis i odczyt danych z plików,
- Narzędzia plikowe

stdlib.h

- Biblioteka standardowych (powszechnych) narzędzi
- Podstawowe operacje matematyczne
- Generator liczb losowych
- Dostęp do funkcji systemowych
- Zarządzanie pamięcią
- Konwersję pomiędzy systemami liczbowymi

math.h

- Biblioteka dostarczająca najczęściej używanych funkcji matematycznych
- Trygonometryczne
- Potęgowe
- Wykładnicze
- Hiperboliczne
- Stałe (e, π)
- Zaokrąglenia

complex.h

- Biblioteka dostarczająca funkcję oraz typ zmiennej zespolonej
- Typ zmiennej zespolonej
- Działania na liczbach zespolonych
- Funkcje zespolone
- Konwersję pomiędzy postaciami liczby zespolonej

stdbool.h

- Biblioteka dostarczająca logicznego typu danych
- Możliwość utworzenia zmiennej typu `bool`
- Dwie predefiniowane stałe
  - `false` – `0` – fałsz
  - `true` – `1` – prawda

time.h

- Dostarcza typy i narzędzia do obsługi czasu i daty
- Nowy typ zmiennej (strukturalnej) pozwalający na przechowywanie informacji o dacie i czasie
- Narzędzia do odczytu i konfiguracji daty i czasu systemowego
- Funkcje przetwarzania daty i czasu

Przykład

- Napisać program demonstrujący możliwości wybranych bibliotek w języku C
- Podzielić kod programu na plik nagłówkowy i źródłowy

Przykład

```
1   #ifndef PROGRAM_H
2   #define PROGRAM_H
3
4   #include <stdio.h>
5   #include <locale.h>
6   #include <stdlib.h>
7   #include <math.h>
8   #include <time.h>
9   #include <complex.h>
10
11  void test_math();
12  void test_complex();
13  void test_random();
14  void pressENTER();
15
16  #endif
```

Przykład

```c
#include "program.h"

int main()
{
    setlocale(LC_CTYPE,"Polish");
    printf("Program testujący wybrane biblioteki języka C\n");
    printf("Naciśnij ENTER...");
    getchar();
    test_math();
    test_complex();
    test_random();
    printf("Koniec dema!");
    return 0;
}
```

```c
void pressENTER()
{
    getchar();
    printf("Naciśnij ENTER...");
    getchar();
}
```

```
E:\Wyk|ady_kod\nowyC\wyk4\program\bin\Debug\program.exe
Program testujący wybrane biblioteki języka C
Naciśnij ENTER...
```

Przykład

```
15  void test_math()
16  {
17      system("cls");
18      printf("Test biblioteki math.h\n");
19      printf("Podaj dowolną liczbę rzeczywistą x=");
20      float x;
21      scanf("%f",&x);
22      printf("sin(%f)=%f\n",x,sin(x));
23      printf("e^%f=%f\n",x,exp(x));
24      printf("ln(%f)=%f\n",x,log(x));
25      printf("log10(%f)=%f\n",x,log10(x));
26      printf("%f^%f=%f\n",x,x,pow(x,x));
27      pressENTER();
28
29  }
```

```
E:\Wyk|ady_kod\nowyC\wyk4\program\bin\Debug\program.exe
Test biblioteki math.h
Podaj dowolną liczbę rzeczywistą x=3.045
sin(3.045000)=0.096442
e^3.045000=21.010033
ln(3.045000)=1.113501
log10(3.045000)=0.483587
3.045000^3.045000=29.684066
Naciśnij ENTER...
```

Przykład

```
30  void test_complex()
31  {
32      system("cls");
33      system("cls");
34      printf("Test biblioteki complex.h\n");
35      float complex Z;
36      float complex Z1 = 10+20*I;
37      float x,y;
38      printf("Podaj liczbę zespoloną Z w postaci algebraicznej (a+bi)=");
39      scanf("%f%fi",&x,&y);
```

E:\Wyk| ady_kod\nowyC\wyk4\program\bin\Debug\program.exe

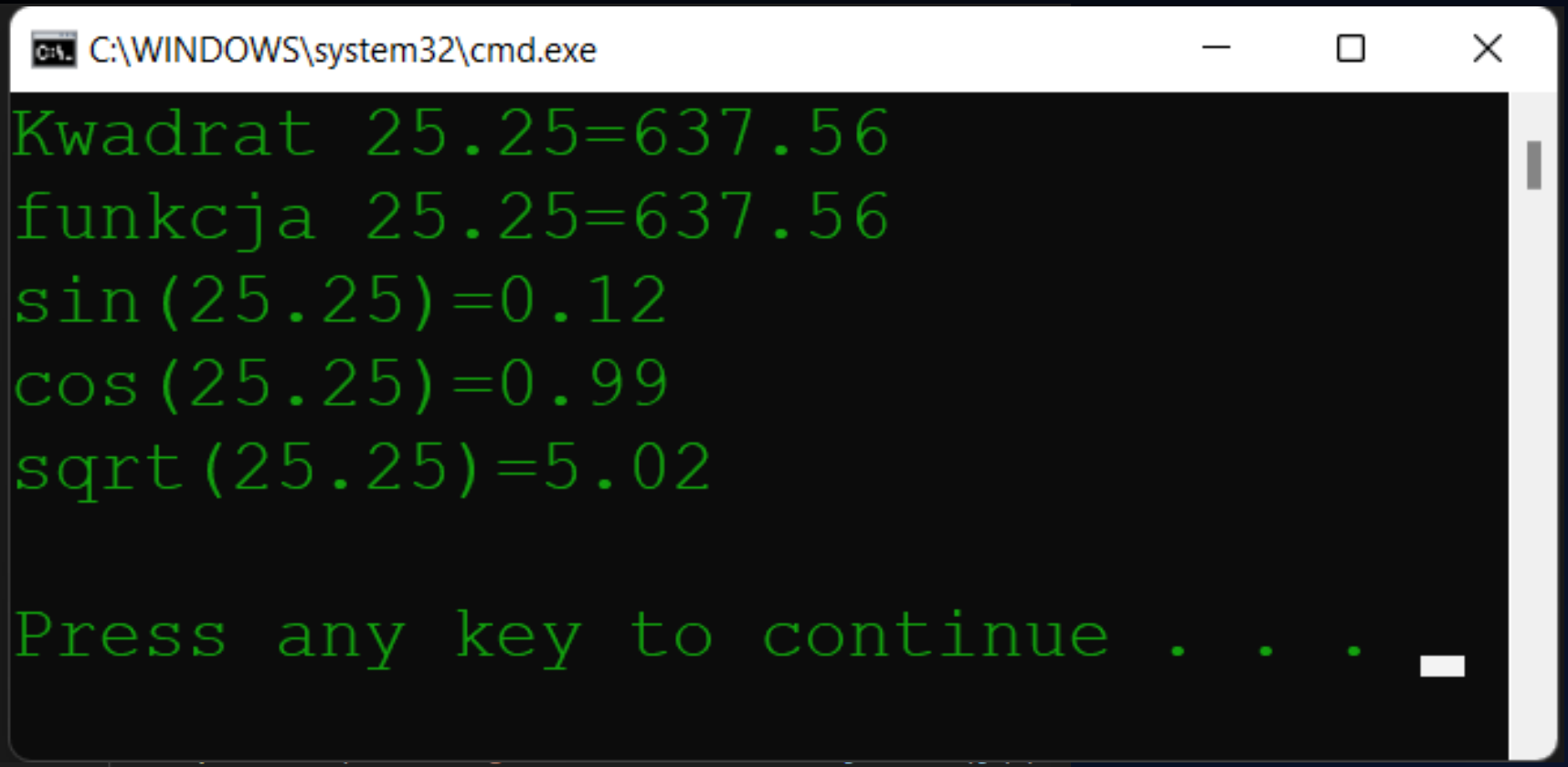
```
Test biblioteki complex.h
Podaj liczbę zespoloną Z w postaci algebraicznej (a+bi)=25+25i
Liczba Z1=10.0+20.0i
Liczba Z=25.0+25.0i
Liczba Z1+Z=35.0+45.0i
Pierwiastek kwadratowy z liczby zespolonej Z=5.49+2.28i
Z^3=-31250.00+31250.00i
Z^Z=-57548099112330632000000000000.00+14151064558769022000000000000.00i
Postać wykładnicza liczby Z = 35.36e^(0.79i)
Postać wykładnicza liczby Z = 35.36e^(45i)
Naciśnij ENTER...
```

Przykład

```
E:\Wyk|ady_kod\nowyC\wyk4\program\bin\Debug\program.exe
Wylosowano liczbę: 11817
Losowanie w przedziale (0,X), podaj X: 10
Wylosowano liczbę: 4
Losowanie w podanym przedziale (X,Y)
Podaj X: -20
Podaj Y: 5
Wylosowano liczbę: 1
Naciśnij ENTER...
```

```
63      printf("Wylosowano liczbę: %i\n", rand()%(X+1));
64      printf("Losowanie w podanym przedziale (X,Y)\n");
65      printf("Podaj X: ");
66      scanf("%i",&X);
67      printf("Podaj Y: ");
68      int Y;
69      scanf("%i",&Y);
70      printf("Wylosowano liczbę: %i\n", X+rand()%(Y-X+1));
71      pressENTER();
72  }
```

Wskaźnik na funkcję




```
15      fun=sin;
16      printf("sin(%.2f)=%.2f\n",y,fun(y));
17      fun=cos;
18      printf("cos(%.2f)=%.2f\n",y,fun(y));
19      fun=sqrt;
20      printf("sqrt(%.2f)=%.2f\n",y,fun(y));
21  }
```

Wskaźnik na funkcję

```c
#include <stdio.h>
#include <math.h>

double fun2(double(*)(double),double,double);
int main()
{
    float x=45, y=60;
    printf("funkcja fun2=%f\n",fun2(sin,x,y));
}
double fun2(double(*f)(double),double a,double b)
{
    return (f(a)+f(b))/2;
}
```

```
C:\WINDOWS\system32\cmd.exe
funkcja fun2=0.273046

Press any key to continue . . .
```

nkcję
etrów

e się

Zadania

1. Napisać funkcję wyświetlającą podaną na wejście liczbę w notacji naukowej x.xxxxExx
2. Napisać program pobierający od użytkownika trzy liczby i obliczający dla nich średnią arytmetyczną i geometryczną. Obliczenia wykonywane są za pomocą funkcji.
3. Program pobiera od użytkownika dwie liczby zespolone i wyświetla na ekranie wynik +,-,* i / tych liczb w postaci algebraicznej i wykładniczej