

Programowanie I. Dane

Aleksander Denisiuk
Uniwersytet Warmińsko-Mazurski
Olsztyn, ul. Słoneczna 54
denisjuk@matman.uwm.edu.pl

24 stycznia 2018

Przechowywanie informacji

Opisanie składni

Zmienne

Operator przypisania

Proste typy danych

Stałe

Ciąg operatorów

Komentarz

Wyjście

Wejście

Szablon Aplikacji

Case Study

Kodowanie

Błędy

Najnowsza wersja tego dokumentu dostępna jest pod adresem

<http://wmii.uwm.edu.pl/~denisjuk/uwm>

Przechowywanie informacji

- ❖ Dane
- ❖ Identyfikator

[Opisanie składni](#)

[Zmienne](#)

[Operator przypisania](#)

[Proste typy danych](#)

[Stałe](#)

[Ciąg operatorów](#)

[Komentarz](#)

[Wyjście](#)

[Wejście](#)

[Szablon Aplikacji](#)

[Case Study](#)

[Kodowanie](#)

[Błędy](#)

Przechowywanie informacji

Dane

Przechowywanie informacji

❖ Dane

❖ Identyfikator

Opisanie składni

Zmienne

Operator przypisania

Proste typy danych

Stałe

Ciąg operatorów

Komentarz

Wyjście

Wejście

Szablon Aplikacji

Case Study

Kodowanie

Błędy

- Informacja, przechowywana w komputerze
- Przetwarzanie
 - ◆ wejście
 - ◆ wyjście
- Zmienna, wartość
- Struktury danych
- Identyfikator
- Deklaracja
- Typ danych

```
int number_of_jars;
```

Identyfikator

Przechowywanie informacji

❖ Dane

❖ Identyfikator

Opisanie składni

Zmienne

Operator przypisania

Proste typy danych

Stałe

Ciąg operatorów

Komentarz

Wyjście

Wejście

Szablon Aplikacji

Case Study

Kodowanie

Błędy

- Nazwa zmiennej (funkcji, klasy, etc) odzwierciadla jej sens
 - ◆ nie używać nazw jednoliterowych
 - ◆ nazwy wielocłonowe łączymy podkreślnikiem: `max_age`
 - ◆ nazwy nie mogą zawierać spacji
 - ◆ nazwy nie mogą zawierać znaków diakrytycznych
 - ◆ używać słów języka angielskiego
 - ◆ nie używać nazw zbyt długich
 - ◆ Niedobre identyfikatory:
`z`, `MAX_age`, `common_number_of_cars`,
`number_of_sold_x`

Konwencje

Przechowywanie informacji

❖ Dane

❖ Identyfikator

Opisanie składni

Zmienne

Operator przypisania

Proste typy danych

Stałe

Ciąg operatorów

Komentarz

Wyjście

Wejście

Szablon Aplikacji

Case Study

Kodowanie

Błędy

- Google C++ Style Guide
- GNU Coding Standards
- NASA C Style Guide
- Etc
 - ◆ styl Pascal: MaxAge
 - ◆ styl Java: maxAge
 - ◆ różne style dla zmiennych, stałych, funkcji, etc

[Przechowywanie informacji](#)

[Opisanie składni](#)

❖ BNF

❖ Diagramy syntaktyczne

[Zmienne](#)

[Operator przypisania](#)

[Proste typy danych](#)

[Stałe](#)

[Ciąg operatorów](#)

[Komentarz](#)

[Wyjście](#)

[Wejście](#)

[Szablon Aplikacji](#)

[Case Study](#)

[Kodowanie](#)

[Błędy](#)

Opisanie składni

Notacja Backusa-Naura

Przechowywanie informacji

Opisanie składni

❖ BNF

❖ Diagramy syntaktyczne

Zmienne

Operator przypisania

Proste typy danych

Stałe

Ciąg operatorów

Komentarz

Wyjście

Wejście

Szablon Aplikacji

Case Study

Kodowanie

Błędy

- Reguły produkcji

$\langle \text{symbol} \rangle ::= \langle \text{wyrażenie zawierające symbole} \rangle$

- metasymbole

◆ \langle

◆ \rangle

◆ $::=$

◆ $|$

- Przykład:

$\langle \text{zero} \rangle ::= 0$

$\langle \text{cyfra niezerowa} \rangle ::= 1|2|3|4|5|6|7|8|9$

$\langle \text{cyfra} \rangle ::= \langle \text{zero} \rangle | \langle \text{cyfra niezerowa} \rangle$

$\langle \text{ciąg cyfr} \rangle ::= \langle \text{cyfra} \rangle | \langle \text{cyfra} \rangle \langle \text{ciąg cyfr} \rangle$

$\langle \text{liczba naturalna} \rangle ::= \langle \text{cyfra} \rangle$

$| \langle \text{cyfra niezerowa} \rangle \langle \text{ciąg cyfr} \rangle$

Rozszerzenia BNF (EBNF)

Przechowywanie informacji

Opisanie składni

❖ BNF

❖ Diagramy syntaktyczne

Zmienne

Operator przypisania

Proste typy danych

Stałe

Ciąg operatorów

Komentarz

Wyjście

Wejście

Szablon Aplikacji

Case Study

Kodowanie

Błędy

- Symbole *terminalne* w cudzysłowie

- Bez $\langle \rangle$

- = zamiast $::=$

- Średnik

- Dodatkowe metasymbole

- ◆ $\{ \}$ — $\{a\}$ zero lub więcej razy a

- ◆ $[]$ — $[a]$ opcjonalne a

zero = "0";

cyfra niezerowa = "1" | "2" | "3" | "4" | "5" | "6"
| "7" | "8" | "9";

cyfra = **zero** | **cyfra niezerowa**;

liczba naturalna = **cyfra niezerowa**, { **cyfra** };

Identyfikator (EBNF)

[Przechowywanie informacji](#)

[Opisanie składni](#)

[❖ BNF](#)

[❖ Diagramy syntaktyczne](#)

[Zmienne](#)

[Operator przypisania](#)

[Proste typy danych](#)

[Stałe](#)

[Ciąg operatorów](#)

[Komentarz](#)

[Wyjście](#)

[Wejście](#)

[Szablon Aplikacji](#)

[Case Study](#)

[Kodowanie](#)

[Błędy](#)

```
letter = "a" | "b" |  
        | "z" | "A" | "B" |  
        | "Z";
```

```
digit = "0" | "1" |  
        | "9";
```

```
id = letter | "_" , { letter | digit | "_" };
```

Diagramy syntaktyczne

[Przechowywanie informacji](#)

[Opisanie składni](#)

❖ BNF

❖ Diagramy syntaktyczne

[Zmienne](#)

[Operator przypisania](#)

[Proste typy danych](#)

[Stałe](#)

[Ciąg operatorów](#)

[Komentarz](#)

[Wyjście](#)

[Wejście](#)

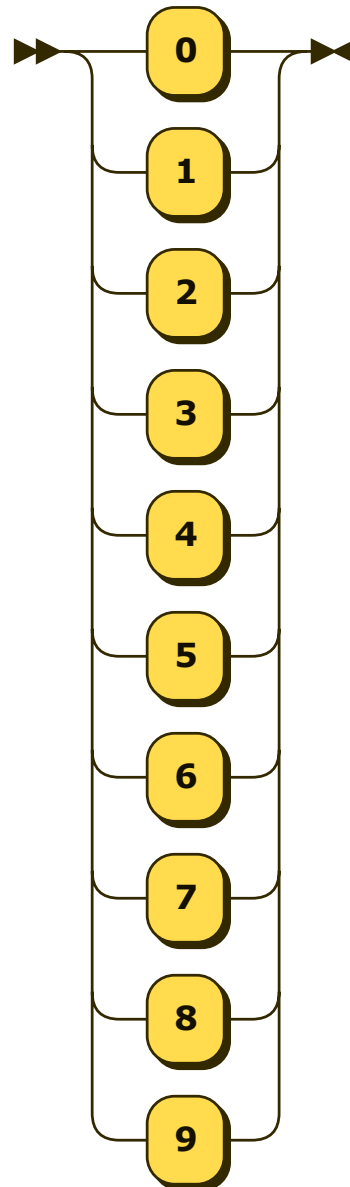
[Szablon Aplikacji](#)

[Case Study](#)

[Kodowanie](#)

[Błędy](#)

● Cyfra:



Identyfikator

[Przechowywanie informacji](#)

[Opisanie składni](#)

❖ BNF

❖ Diagramy syntaktyczne

[Zmienne](#)

[Operator przypisania](#)

[Proste typy danych](#)

[Stałe](#)

[Ciąg operatorów](#)

[Komentarz](#)

[Wyjście](#)

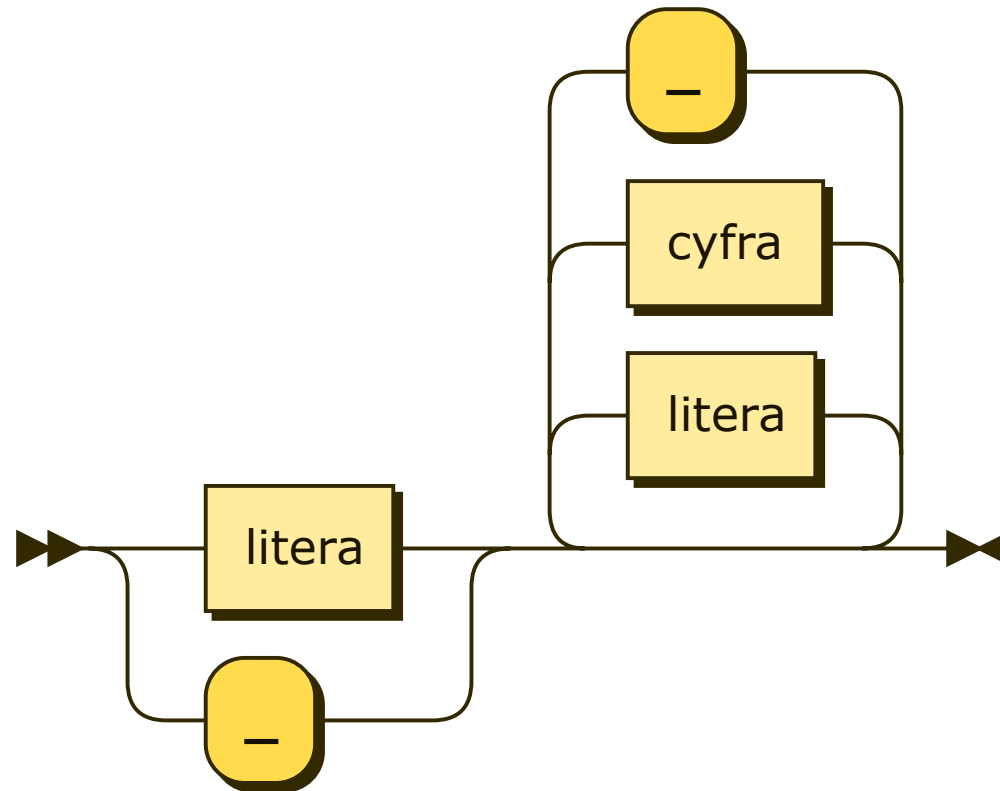
[Wejście](#)

[Szablon Aplikacji](#)

[Case Study](#)

[Kodowanie](#)

[Błędy](#)



[Przechowywanie informacji](#)

[Opisanie składni](#)

[Zmienne](#)

[❖ Deklaracja](#)

[Operator przypisania](#)

[Proste typy danych](#)

[Stałe](#)

[Ciąg operatorów](#)

[Komentarz](#)

[Wyjście](#)

[Wejście](#)

[Szablon Aplikacji](#)

[Case Study](#)

[Kodowanie](#)

[Błędy](#)

Zmienne

Deklaracja zmiennej

Przechowywanie informacji

Opisanie składni

Zmienne

❖ Deklaracja

Operator przypisania

Proste typy danych

Stałe

Ciąg operatorów

Komentarz

Wyjście

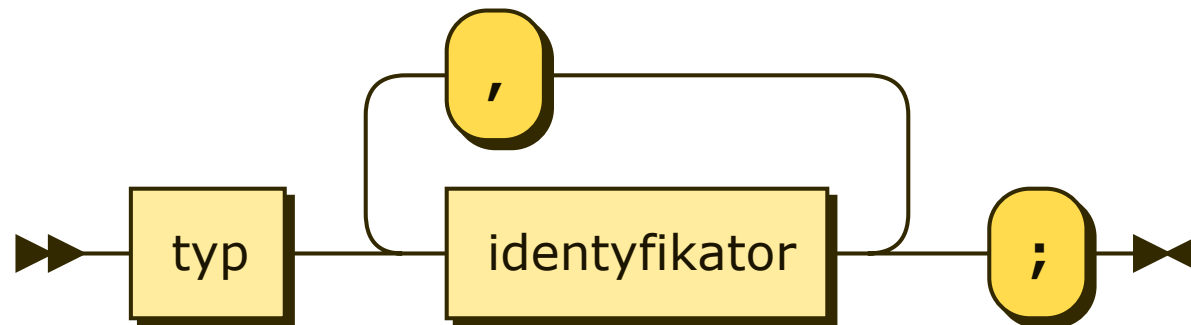
Wejście

Szablon Aplikacji

Case Study

Kodowanie

Błędy



● Przykłady:

```
int n_of_tests,  
      curr_test;  
double width, height;  
unsigned int age;  
char name;
```

Semantyka

Przechowywanie informacji

Opisanie składni

Zmienne

❖ Deklaracja

Operator przypisania

Proste typy danych

Stałe

Ciąg operatorów

Komentarz

Wyjście

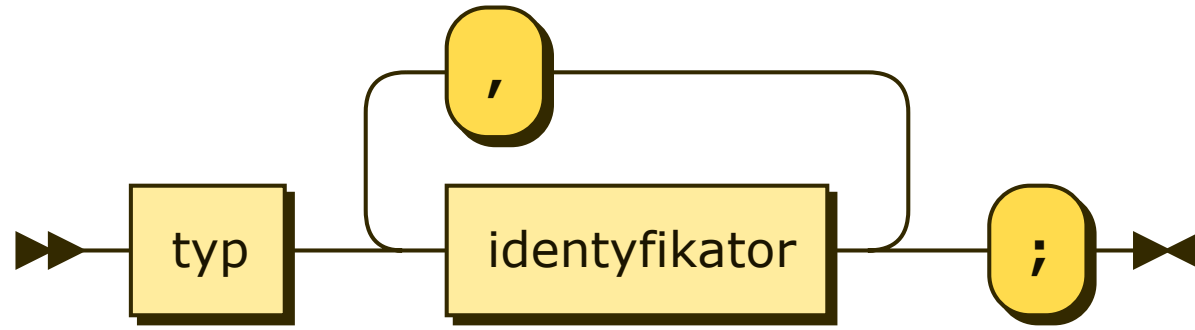
Wejście

Szablon Aplikacji

Case Study

Kodowanie

Błędy



- *Białe znaki* są ignorowane
- Każdy identyfikator opisują oddzielną zmienną tego samego typu
- Każdy identyfikator opisuje zmienną tylko jeden raz w bieżącej *jednostce programowej*
- Niektóre identyfikatory są *zarezerwowane*: **int**, **double**, **for**, **return**, ...

[Przechowywanie informacji](#)

[Opisanie składni](#)

[Zmienne](#)

[Operator przypisania](#)

❖ [Operator przypisania](#)

[Proste typy danych](#)

[Stałe](#)

[Ciąg operatorów](#)

[Komentarz](#)

[Wyjście](#)

[Wejście](#)

[Szablon Aplikacji](#)

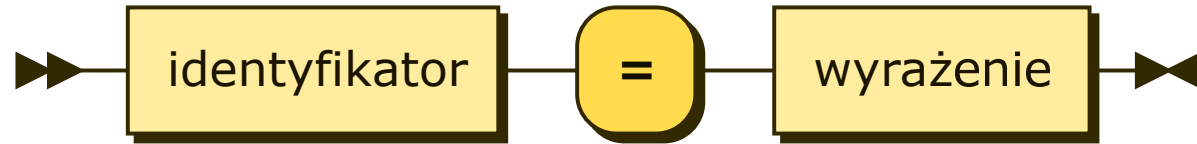
[Case Study](#)

[Kodowanie](#)

[Błędy](#)

Operator przypisania

Operator przypisania



- Zmiennej, która jest określana przez identyfikator, przypisywana wartość wyrażenia
- Identyfikator powinien być wcześniej zadeklarowany jako zmienna
- Typy zmiennej i wyrażenia powinny być zgodne
- Przykłady:

```
int n_of_jugs;
```

```
double temperature;
```

```
n_of_jugs = 0;
```

```
temperature = -36.6;
```

Przechowywanie informacji

Opisanie składni

Zmienne

Operator przypisania

❖ Operator przypisania

Proste typy danych

Stałe

Ciąg operatorów

Komentarz

Wyjście

Wejście

Szablon Aplikacji

Case Study

Kodowanie

Błędy

Przypisanie w momencie deklaracji

Przechowywanie informacji

Opisanie składni

Zmienne

Operator przypisania

❖ Operator przypisania

Proste typy danych

Stałe

Ciąg operatorów

Komentarz

Wyjście

Wejście

Szablon Aplikacji

Case Study

Kodowanie

Błędy

```
int n_of_jugs = 0;  
double temperature = -36.6;
```

Rzutowanie (konwersja typów)

Przechowywanie informacji

Opisanie składni

Zmienne

Operator przypisania

❖ Operator przypisania

Proste typy danych

Stałe

Ciąg operatorów

Komentarz

Wyjście

Wejście

Szablon Aplikacji

Case Study

Kodowanie

Błędy

- Odrzucenie bitów (dopisanie
- Z liczb rzeczywistych na liczby naturalne — odrzucenie części ułamkowej
- Konwersja niejawna

```
int n = 2.7828;  
sqrt(2);
```

- Konwersja jawna

```
(int) (-3.1415);  
sqrt((double)2);
```

[Przechowywanie informacji](#)

[Opisanie składni](#)

[Zmienne](#)

[Operator przypisania](#)

[Proste typy danych](#)

❖ Typy danych

❖ Liczby całkowite

❖ Liczby rzeczywiste

❖ Litery

[Stałe](#)

[Ciąg operatorów](#)

[Komentarz](#)

[Wyjście](#)

[Wejście](#)

[Szablon Aplikacji](#)

[Case Study](#)

[Kodowanie](#)

[Błędy](#)

Proste typy danych

Typy danych

[Przechowywanie informacji](#)

[Opisanie składni](#)

[Zmienne](#)

[Operator przypisania](#)

[Proste typy danych](#)

❖ Typy danych

❖ Liczby całkowite

❖ Liczby rzeczywiste

❖ Litery

[Stałe](#)

[Ciąg operatorów](#)

[Komentarz](#)

[Wyjście](#)

[Wejście](#)

[Szablon Aplikacji](#)

[Case Study](#)

[Kodowanie](#)

[Błędy](#)

- Proste
- Złożone
 - ◆ Identyfikator typu
 - ◆ Zbiór wartości
 - ◆ Działania na danych tego typu

Liczby całkowite

Przechowywanie informacji

Opisanie składni

Zmienne

Operator przypisania

Proste typy danych

❖ Typy danych

❖ **Liczby całkowite**

❖ Liczby rzeczywiste

❖ Litery

Stałe

Ciąg operatorów

Komentarz

Wyjście

Wejście

Szablon Aplikacji

Case Study

Kodowanie

Błędy

- **int**
- Kwalifikatory
 - ◆ **long/short**
 - ◆ **signed/unsigned**

- Przykłady:

```
int n_of_jugs;
```

```
long int n_of_jugs;
```

```
unsigned long int n_of_jugs;
```

```
long long int very_big;
```

Wartości

Przechowywanie informacji

Opisanie składni

Zmienne

Operator przypisania

Proste typy danych

❖ Typy danych

❖ Liczby całkowite

❖ Liczby rzeczywiste

❖ Litery

Stałe

Ciąg operatorów

Komentarz

Wyjście

Wejście

Szablon Aplikacji

Case Study

Kodowanie

Błędy

- **int** ma minimum 16 bitów
- **long int** jest co najmniej takiego rozmiaru, co **int**
- **long long int** ma minimum 64 bitów
- Dziesiętne: 0 1 -2
- Ósemkowe: 01 013 -025
 - ◆ 0178 (błąd!)
- Szesnastkowe: 0x10 0x11 0xff

Operacje

Przechowywanie informacji

Opisanie składni

Zmienne

Operator przypisania

Proste typy danych

❖ Typy danych

❖ Liczby całkowite

❖ Liczby rzeczywiste

❖ Litery

Stałe

Ciąg operatorów

Komentarz

Wyjście

Wejście

Szablon Aplikacji

Case Study

Kodowanie

Błędy

● Dwuargumentowe:

◆ Dodawanie: $x + y$

◆ Odejmowanie: $x - y$

◆ Mnożenie: $x * y$

◆ Dzielenie: x / y (część ułamkowa się odrzuca (nie zaokrąglenie (może być w stronę zwiększenia dla ujemnych liczb)))

◆ Dzielenie: $x \% y$ (reszta przy dzieleniu)

● Jednoargumentowe:

◆ Plus: $+$ y

◆ Minus: $-$ y

Wyrażenia

Przechowywanie informacji

Opisanie składni

Zmienne

Operator przypisania

Proste typy danych

❖ Typy danych

❖ Liczby całkowite

❖ Liczby rzeczywiste

❖ Litery

Stałe

Ciąg operatorów

Komentarz

Wyjście

Wejście

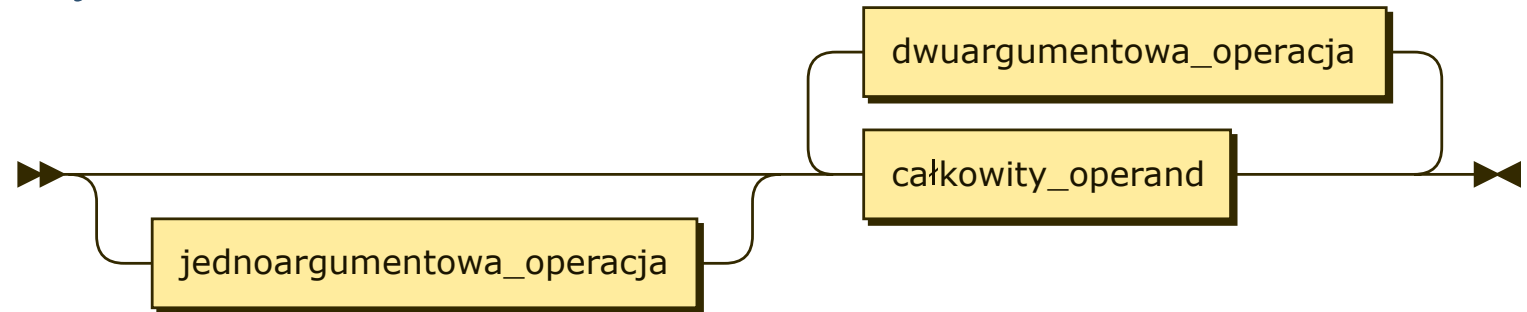
Szablon Aplikacji

Case Study

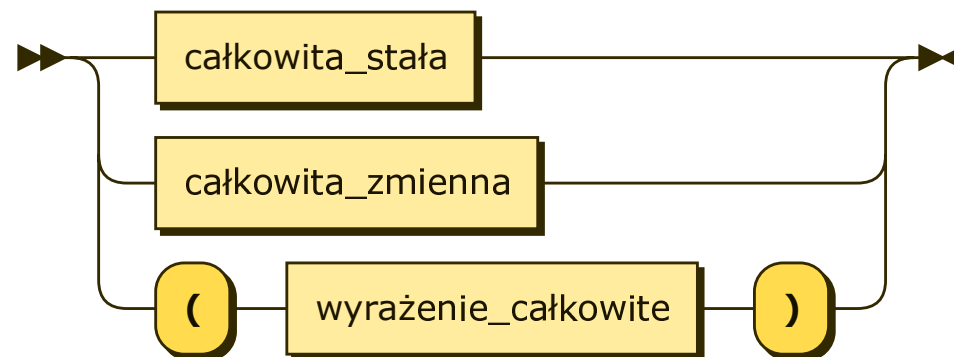
Kodowanie

Błędy

● Wyrażenie całkowite:



● Całkowity operand:



$(-1 + 2 * (3 - 4 * (6 - 3) / 2 / 3) * 7) - 8$

Operatory inkrementacji i dekrementacji

[Przechowywanie informacji](#)

[Opisanie składni](#)

[Zmienne](#)

[Operator przypisania](#)

[Proste typy danych](#)

❖ [Typy danych](#)

❖ [Liczby całkowite](#)

❖ [Liczby rzeczywiste](#)

❖ [Litery](#)

[Stałe](#)

[Ciąg operatorów](#)

[Komentarz](#)

[Wyjście](#)

[Wejście](#)

[Szablon Aplikacji](#)

[Case Study](#)

[Kodowanie](#)

[Błędy](#)

- ++ oraz --
- Tylko przy zmiennych
- prefiksowy

```
n = 5;
```

```
x = ++n;
```

- postfiksowy

```
n = 5;
```

```
x = n++;
```

Liczby rzeczywiste

Przechowywanie informacji

Opisanie składni

Zmienne

Operator przypisania

Proste typy danych

❖ Typy danych

❖ Liczby całkowite

❖ **Liczby rzeczywiste**

❖ Litery

Stałe

Ciąg operatorów

Komentarz

Wyjście

Wejście

Szablon Aplikacji

Case Study

Kodowanie

Błędy

- **float, double, long double**

- Przykłady:

```
float x_coord;
```

```
double distance;
```

```
long double population;
```

- ◆ co jest nie tak w ostatnim przykładzie?

Wartości

Przechowywanie informacji

Opisanie składni

Zmienne

Operator przypisania

Proste typy danych

❖ Typy danych

❖ Liczby całkowite

❖ Liczby rzeczywiste

❖ Litery

Stałe

Ciąg operatorów

Komentarz

Wyjście

Wejście

Szablon Aplikacji

Case Study

Kodowanie

Błędy

- **float** ma co najmniej 4 bajty
- **double** — 8 bajtów
- **long double** zazwyczaj ma 8-16 bajtów
- Stałe:
 - ◆ **double**: `0.178`, `-3.1415`
 - ◆ **float**: `0.178f`, `-3.1415f`
 - ◆ **long double**:
`3.14159265358979323846264338328L`
 - ◆ notacja wykładnicza: `1.23e+17`, `0.45e-2`
- Liczby *zmiennoprzecinkowe*

Arytmetyka zmiennoprzecinkowa

Przechowywanie informacji

Opisanie składni

Zmienne

Operator przypisania

Proste typy danych

❖ Typy danych

❖ Liczby całkowite

❖ Liczby rzeczywiste

❖ Litery

Stałe

Ciąg operatorów

Komentarz

Wyjście

Wejście

Szablon Aplikacji

Case Study

Kodowanie

Błędy

● Zaokrąglenie

◆ $(x + y) + z \neq x + (y + z)$

◆ $(x * y) * z \neq x * (y * z)$

◆ $(x + y) * z \neq x * z + y * z$

● Biblioteka `math.h`

◆ `NAN`

Litery

[Przechowywanie informacji](#)

[Opisanie składni](#)

[Zmienne](#)

[Operator przypisania](#)

[Proste typy danych](#)

❖ [Typy danych](#)

❖ [Liczby całkowite](#)

❖ [Liczby rzeczywiste](#)

❖ [Litery](#)

[Stałe](#)

[Ciąg operatorów](#)

[Komentarz](#)

[Wyjście](#)

[Wejście](#)

[Szablon Aplikacji](#)

[Case Study](#)

[Kodowanie](#)

[Błędy](#)

- **char**
- **signed char**
- Przykłady:

char first_letter;
- Typ liczbowy (całkowity)
- Działania arytmetyczne, jak dla liczb całkowitych

Wartości

Przechowywanie informacji

Opisanie składni

Zmienne

Operator przypisania

Proste typy danych

❖ Typy danych

❖ Liczby całkowite

❖ Liczby rzeczywiste

❖ Litery

Stałe

Ciąg operatorów

Komentarz

Wyjście

Wejście

Szablon Aplikacji

Case Study

Kodowanie

Błędy

- Co najmniej jeden bajt

```
char letter = 'A';  
letter = ' ';  
letter = 32;  
letter = '\040';  
letter = '\x20';  
letter = '\t';
```

- Niektóre wartości specjalne

```
"\t \n \\ \' \" \0 \ooo \xhh"
```

- Polskie znaki w kodowaniu Unicode (UTF-8): mogą być problemy

◆ wchar

Stałe tekstowe

[Przechowywanie informacji](#)

[Opisanie składni](#)

[Zmienne](#)

[Operator przypisania](#)

[Proste typy danych](#)

❖ [Typy danych](#)

❖ [Liczby całkowite](#)

❖ [Liczby rzeczywiste](#)

❖ [Litery](#)

[Stałe](#)

[Ciąg operatorów](#)

[Komentarz](#)

[Wyjście](#)

[Wejście](#)

[Szablon Aplikacji](#)

[Case Study](#)

[Kodowanie](#)

[Błędy](#)

- Stałe tekstowe w cudzysłowie
" ", "a", "dwa razy", "Zażółć gęślą jaźń"
- Unicode (UTF-8): nie powinno być problemu

[Przechowywanie informacji](#)

[Opisanie składni](#)

[Zmienne](#)

[Operator przypisania](#)

[Proste typy danych](#)

[Stałe](#)

❖ Stałe

❖ Makrodefinicje

[Ciąg operatorów](#)

[Komentarz](#)

[Wyjście](#)

[Wejście](#)

[Szablon Aplikacji](#)

[Case Study](#)

[Kodowanie](#)

[Błędy](#)

Stałe

Stałe

Przechowywanie informacji

Opisanie składni

Zmienne

Operator przypisania

Proste typy danych

Stałe

❖ Stałe

❖ Makrodefinicje

Ciąg operatorów

Komentarz

Wyjście

Wejście

Szablon Aplikacji

Case Study

Kodowanie

Błędy

- Modyfikator **const**
- Przykłady:

```
const int kMaxNumber = 1024;  
const double kDegreesToRadians  
= 3.141592653589793/180.0;
```

Makrodefinicje

Przechowywanie informacji

Opisanie składni

Zmienne

Operator przypisania

Proste typy danych

Stałe

❖ Stałe

❖ Makrodefinicje

Ciąg operatorów

Komentarz

Wyjście

Wejście

Szablon Aplikacji

Case Study

Kodowanie

Błędy

- Konstrukcja *#define*
- Preprocesor
- Przykłady:

```
#define MAX_NUMBER 1024
```

```
#define DEGREES_TO_RADIANS (3.1415/180.0)
```

- Uwaga:

```
deg = rad / DEGREES_TO_RADIANS;
```

[Przechowywanie informacji](#)

[Opisanie składni](#)

[Zmienne](#)

[Operator przypisania](#)

[Proste typy danych](#)

[Stałe](#)

[Ciąg operatorów](#)

❖ Ciąg operatorów

[Komentarz](#)

[Wyjście](#)

[Wejście](#)

[Szablon Aplikacji](#)

[Case Study](#)

[Kodowanie](#)

[Błędy](#)

Ciąg operatorów

Ciąg operatorów

Przechowywanie informacji

Opisanie składni

Zmienne

Operator przypisania

Proste typy danych

Stałe

Ciąg operatorów

❖ Ciąg operatorów

Komentarz

Wyjście

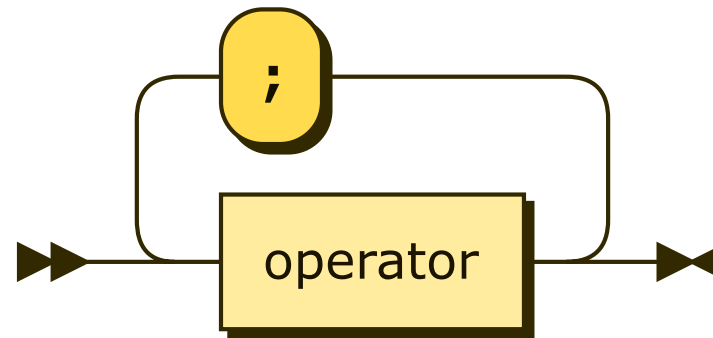
Wejście

Szablon Aplikacji

Case Study

Kodowanie

Błędy



- Jeden operator w wierszu
- Początek w tej samej pozycji
- Każda zmienna powinna być *zainicjalizowana*
- Przykład:

```
curr_jug = 0;  
empty_jugs = empty_jugs + 1;  
curr_jug = 10;
```

Zadanie

[Przechowywanie informacji](#)

[Opisanie składni](#)

[Zmienne](#)

[Operator przypisania](#)

[Proste typy danych](#)

[Stałe](#)

[Ciąg operatorów](#)

❖ Ciąg operatorów

[Komentarz](#)

[Wyjście](#)

[Wejście](#)

[Szablon Aplikacji](#)

[Case Study](#)

[Kodowanie](#)

[Błędy](#)

- Zamienić miejscami wartości zmiennych `width` i `height`
 - ❖ `width ↔ height`

[Przechowywanie informacji](#)

[Opisanie składni](#)

[Zmienne](#)

[Operator przypisania](#)

[Proste typy danych](#)

[Stałe](#)

[Ciąg operatorów](#)

[Komentarz](#)

❖ [Komentarze](#)

[Wyjście](#)

[Wejście](#)

[Szablon Aplikacji](#)

[Case Study](#)

[Kodowanie](#)

[Błędy](#)

Komentarz

Komentarze

Przechowywanie informacji

Opisanie składni

Zmienne

Operator przypisania

Proste typy danych

Stałe

Ciąg operatorów

Komentarz

❖ Komentarze

Wyjście

Wejście

Szablon Aplikacji

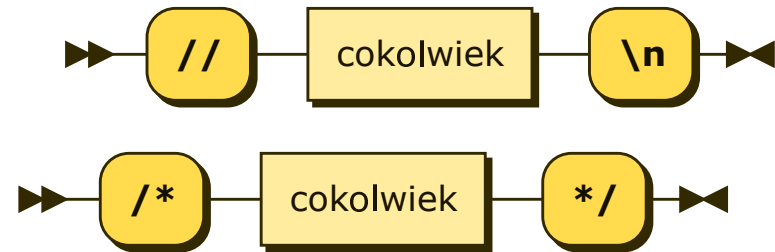
Case Study

Kodowanie

Błędy

- Do końca wiersza:

- Wielowierszowy:



Wykorzystanie komentarzy

Przechowywanie informacji

Opisanie składni

Zmienne

Operator przypisania

Proste typy danych

Stałe

Ciąg operatorów

Komentarz

❖ Komentarze

Wyjście

Wejście

Szablon Aplikacji

Case Study

Kodowanie

Błędy

- Dużo komentarzy:
 - ✦ na przyszłość
 - ✦ dla innych
 - ✦ około 30%
- Nie dużo komentarzy:
 - ✦ nie komentować oczywistych miejsc
- Obok tego miejsca, które się komentuje
 - ✦ takie same wcięcie
- Opisanie identyfikatorów
- Trudne fragmenty programu
- Na początku dużego bloku kodu
- W miejscach kluczowych

Przykłady komentarzy

Przechowywanie informacji

Opisanie składni

Zmienne

Operator przypisania

Proste typy danych

Stałe

Ciąg operatorów

Komentarz

❖ Komentarze

Wyjście

Wejście

Szablon Aplikacji

Case Study

Kodowanie

Błędy

```
int pigs; // ilość świń na 1 stycznia
```

```
int pig_pens; // ilość chlewów
```

```
int new_pens; // ilość chlewów nie oddanych
```

```
pig_pens = 5;
```

```
new_pens = ( pig_pens + 3 ) * ( 2 + 1 );
```

```
/* Stwierdzenie:
```

```
    pig_pens == 20, new_pens == 24
```

```
*/
```

[Przechowywanie informacji](#)

[Opisanie składni](#)

[Zmienne](#)

[Operator przypisania](#)

[Proste typy danych](#)

[Stałe](#)

[Ciąg operatorów](#)

[Komentarz](#)

[Wyjście](#)

❖ Funkcja `printf`

❖ Format

[Wejście](#)

[Szablon Aplikacji](#)

[Case Study](#)

[Kodowanie](#)

[Błędy](#)

Wyjście

Funkcja *printf*

Przechowywanie informacji

Opisanie składni

Zmienne

Operator przypisania

Proste typy danych

Stałe

Ciąg operatorów

Komentarz

Wyjście

❖ Funkcja `printf`

❖ Format

Wejście

Szablon Aplikacji

Case Study

Kodowanie

Błędy

- Brak standardowego operatora
- Biblioteka `stdio.h`

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

- Funkcja `printf`

```
printf(format, dane);
```

- ◆ `format` — stała tekstowa
- ◆ `dane` — zero lub więcej wyrażeń, rozdzielonych przecinkiem

Format

Przechowywanie informacji

Opisanie składni

Zmienne

Operator przypisania

Proste typy danych

Stałe

Ciąg operatorów

Komentarz

Wyjście

❖ Funkcja `printf`

❖ Format

Wejście

Szablon Aplikacji

Case Study

Kodowanie

Błędy

- Tekst oraz sekwencje sterujące, poprzedzone znakiem %
- Niektóre sekwencje sterujące:
 - ◆ `%d` lub `%i` — liczba całkowita
 - ◆ `%u` — dodatnia liczba całkowita (**unsigned**)
 - ◆ `%f`, `%e`, `%g` — liczba rzeczywista
 - ◆ `%ld` — liczba całkowita typu **long int**
 - ◆ `%lf` — liczba rzeczywista typu **double**
 - ◆ `%c` — litera
 - ◆ `%s` — tekst

Przykłady

Przechowywanie informacji

Opisanie składni

Zmienne

Operator przypisania

Proste typy danych

Stałe

Ciąg operatorów

Komentarz

Wyjście

❖ Funkcja `printf`

❖ Format

Wejście

Szablon Aplikacji

Case Study

Kodowanie

Błędy

● Kod:

```
printf ("Witaj Świecie!\n");  
printf ("Litera: %c %c \n", 'a', 65);  
printf ("Liczby: %d %ld\n", 1977, 650000L);  
printf ("floats: %4.2f %+0e %E \n", 3.1416,  
        3.1416, 3.1416  
        );  
printf ("%s \n", "Długi tekst");
```

● Wyjście:

```
Witaj Świecie!  
Litera: a A  
Liczby: 1977 650000  
floats: 3.14 +3e+000 3.141600E+000  
Długi tekst
```

[Przechowywanie informacji](#)

[Opisanie składni](#)

[Zmienne](#)

[Operator przypisania](#)

[Proste typy danych](#)

[Stałe](#)

[Ciąg operatorów](#)

[Komentarz](#)

[Wyjście](#)

[Wejście](#)

❖ Funkcja `scanf`

❖ Format

[Szablon Aplikacji](#)

[Case Study](#)

[Kodowanie](#)

[Błędy](#)

Wejście

Funkcja *scanf*

Przechowywanie informacji

Opisanie składni

Zmienne

Operator przypisania

Proste typy danych

Stałe

Ciąg operatorów

Komentarz

Wyjście

Wejście

❖ Funkcja `scanf`

❖ Format

Szablon Aplikacji

Case Study

Kodowanie

Błędy

- Brak standardowego operatora

- Biblioteka `stdio.h`

`#include <stdio.h>`

- Funkcja `scanf`

`scanf(format, dane);`

◆ `format` — stała tekstowa

◆ `dane` — jedna lub więcej zmiennych, rozdzielonych przecinkiem i poprzedzonych znakiem `&`

Format

Przechowywanie informacji

Opisanie składni

Zmienne

Operator przypisania

Proste typy danych

Stałe

Ciąg operatorów

Komentarz

Wyjście

Wejście

❖ Funkcja `scanf`

❖ Format

Szablon Aplikacji

Case Study

Kodowanie

Błędy

- Niektóre sekwencje sterujące:
 - ◆ `%i` — liczba całkowita
 - ◆ `%d` — *dziesiętna* liczba całkowita
 - ◆ `%u` — dodatnia dziesiętna liczba całkowita
 - ◆ `%f`, `%e`, `%g` — liczba rzeczywista
 - ◆ `%ld` — liczba całkowita typu **long int**
 - ◆ `%lf` — liczba rzeczywista typu **double**
 - ◆ `%c` — litera

Przykłady

Przechowywanie informacji

Opisanie składni

Zmienne

Operator przypisania

Proste typy danych

Stałe

Ciąg operatorów

Komentarz

Wyjście

Wejście

❖ Funkcja `scanf`

❖ Format

Szablon Aplikacji

Case Study

Kodowanie

Błędy

● Kod:

```
int weight;  
double size;
```

```
printf ("Podaj swój wzrost w metrach: ");  
scanf ("%lf",&size);  
printf ("Podaj swoją wagę w kg: ");  
scanf ("%d",&weight);  
printf ("Twój indeks BMI: %f\n",  
        weight/(size*size) );
```

● Wyjście:

```
Podaj swój wzrost w metrach: 1.84  
Podaj swoją wagę w kg: 86  
Twój indeks BMI: 25.10633
```

[Przechowywanie informacji](#)

[Opisanie składni](#)

[Zmienne](#)

[Operator przypisania](#)

[Proste typy danych](#)

[Stałe](#)

[Ciąg operatorów](#)

[Komentarz](#)

[Wyjście](#)

[Wejście](#)

[Szablon Aplikacji](#)

❖ Szablon

[Case Study](#)

[Kodowanie](#)

[Błędy](#)

Szablon Aplikacji

Prosta aplikacja

Przechowywanie informacji

Opisanie składni

Zmienne

Operator przypisania

Proste typy danych

Stałe

Ciąg operatorów

Komentarz

Wyjście

Wejście

Szablon Aplikacji

❖ Szablon

Case Study

Kodowanie

Błędy

```
/* ****  
 * Ten program wprowadza liczbę -- wagę  
 * w kilogramach i wyświetla odpowiednią  
 * wagę w funtach, zaokrągloną do całkowitej  
 **** */  
  
#include <stdio.h>
```

```
int main(void) {  
    unsigned int kg; //waga w kilogramach  
    unsigned int pounds; // waga w funtach  
    printf("Wprowadź swoją wagę w kg: ");  
    scanf("%u", &kg);  
    pounds = 2.20462262185*kg + 0.5;  
    printf("Waga w funtach: %u\n", pounds);  
    printf("(Zaokrąglono do całkowitej)\n");  
    return 0;  
}
```

Formatowanie kodu

Przechowywanie informacji

Opisanie składni

Zmienne

Operator przypisania

Proste typy danych

Stałe

Ciąg operatorów

Komentarz

Wyjście

Wejście

Szablon Aplikacji

❖ Szablon

Case Study

Kodowanie

Błędy

- Na początku umieszcza się opisanie programu
 - ✦ można dodać licencję
- Każdy blok kodu w nawiasie klamrowym otrzymuje wcięcie o dodatkowe 3 spacje
- Logiczne bloki kodu oddzielamy pustą linią
 - ✦ można dodać pustą linię po deklaracji zmiennych
- Zmienne nie oczywiste należy skomentować
 - ✦ Jeżeli zmiennych jest dużo, uporządkować alfabetycznie
- Stwierdzenia dostają takie same wcięcie

```
pounds = 2.20462262185*kg + 0.5;  
/* Stwierdzenie: zmienna pound zawiera  
 * odpowiednią zmiennej kg wagę w funtach  
 */
```

[Przechowywanie informacji](#)

[Opisanie składni](#)

[Zmienne](#)

[Operator przypisania](#)

[Proste typy danych](#)

[Stałe](#)

[Ciąg operatorów](#)

[Komentarz](#)

[Wyjście](#)

[Wejście](#)

[Szablon Aplikacji](#)

Case Study

❖ Sformułowanie

[Kodowanie](#)

[Błędy](#)

Case Study

Przechowywanie informacji

Opisanie składni

Zmienne

Operator przypisania

Proste typy danych

Stałe

Ciąg operatorów

Komentarz

Wyjście

Wejście

Szablon Aplikacji

Case Study

❖ Sformułowanie

Kodowanie

Błędy

Nazwa

Wyświetlenie trzech liter w odwrotnej kolejności

Opisanie

Program wprowadza cztery symbole i wyświetla je w odwrotnej kolejności

Dane Wejściowe

Wprowadza się cztery symbole w jednej linijce.

Dane Wyjściowe

Na początku wyświetla się zachęta dla użytkownika:

`Wprowadź cztery symbole (w jednej linijce)`

Po wprowadzeniu przez użytkownika symboli wyświetla się pusta linia, a potem w jednej linijce cztery wprowadzone symbole w kolejności, odwrotnej do kolejności wprowadzenia.

Błędne dane i przykład

Przechowywanie informacji

Opisanie składni

Zmienne

Operator przypisania

Proste typy danych

Stałe

Ciąg operatorów

Komentarz

Wyjście

Wejście

Szablon Aplikacji

Case Study

❖ Sformułowanie

Kodowanie

Błędy

Błędne Dane

1. Jeżeli wprowadzono mniej niż cztery liczby, program czeka na kontynuację wejścia
2. Wszystkie wprowadzone dane, oprócz pierwszych czterech symboli, są ignorowane

Przykład

Wprowadź cztery symbole (w jednej linii)

EDIT

Twoje symbole w odwrotnej kolejności: TIDE

Rozwiązanie

Przechowywanie informacji

Opisanie składni

Zmienne

Operator przypisania

Proste typy danych

Stałe

Ciąg operatorów

Komentarz

Wyjście

Wejście

Szablon Aplikacji

Case Study

❖ Sformułowanie

Kodowanie

Błędy

```
/* ****  
 * Ten program wprowadza cztery symbole  
 * a potem je wyświetla w odwrotnej kolejności  
 *  
 **** */  
  
#include <stdio.h>
```

```
int main(void) {  
    char symbol1, symbol2, symbol3, symbol4;  
    printf("Wprowadź cztery symbole (w jednej linii): ");  
    scanf("%c%c%c%c", &symbol1, &symbol2, &symbol3, &symbol4);  
    printf("\nTwoje symbole: %c%c%c%c\n", symbol4, symbol3, symbol2, symbol1);  
    return 0;  
}
```

Formatowanie długich wierszy

Przechowywanie informacji

Opisanie składni

Zmienne

Operator przypisania

Proste typy danych

Stałe

Ciąg operatorów

Komentarz

Wyjście

Wejście

Szablon Aplikacji

Case Study

❖ Sformułowanie

Kodowanie

Błędy

```
printf("Wprowadź cztery symbole\  
(w jednej linii) \n");  
scanf("%c%c%c%c",  
      &symbol1,  
      &symbol2,  
      &symbol3,  
      &symbol4  
);  
printf("\nTwoje symbole: %c%c%c%c\n",  
      symbol4,  
      symbol3,  
      symbol2,  
      symbol1  
);
```

[Przechowywanie informacji](#)

[Opisanie składni](#)

[Zmienne](#)

[Operator przypisania](#)

[Proste typy danych](#)

[Stałe](#)

[Ciąg operatorów](#)

[Komentarz](#)

[Wyjście](#)

[Wejście](#)

[Szablon Aplikacji](#)

[Case Study](#)

Kodowanie

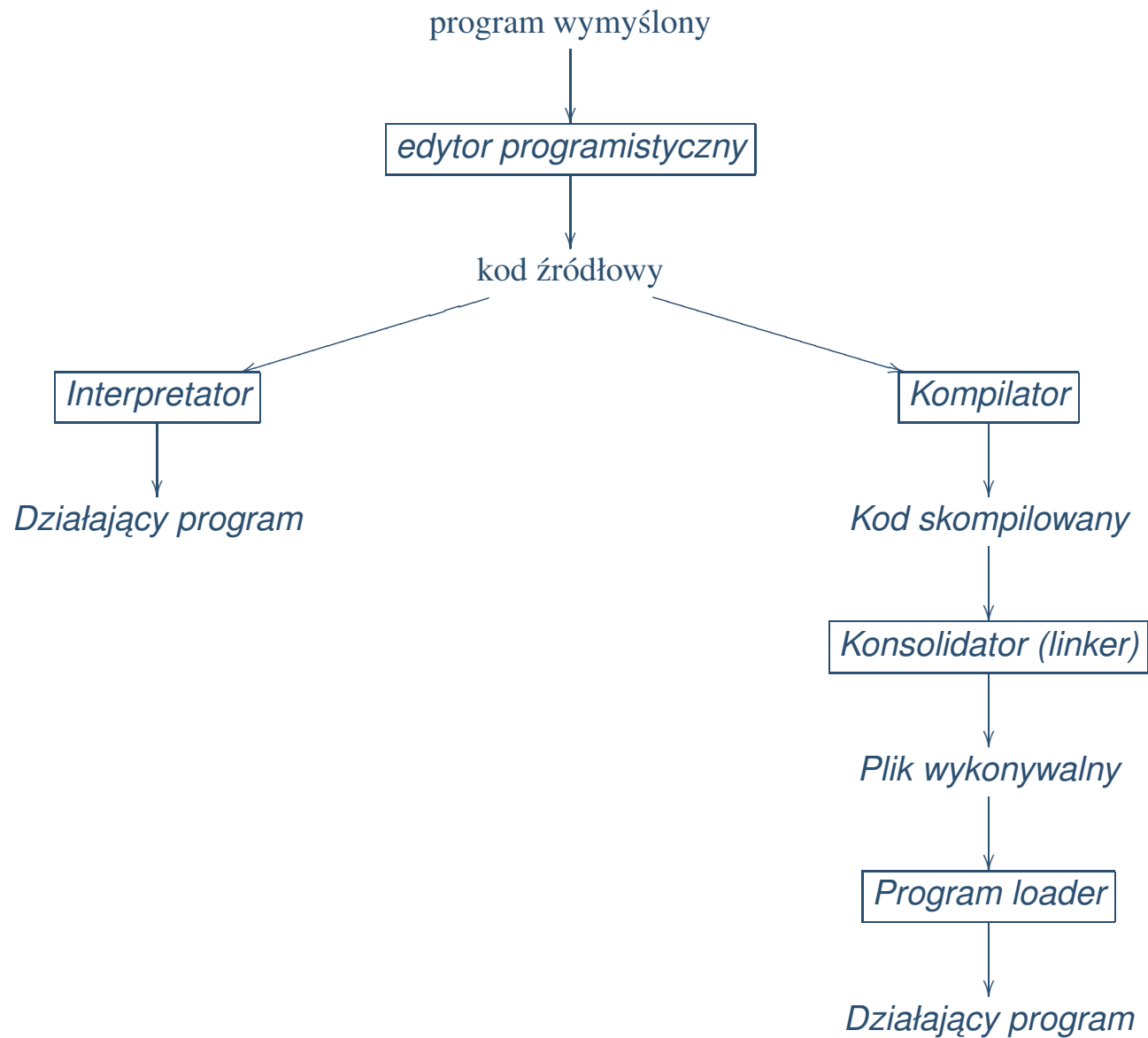
❖ Proces programowania

❖ Makefile

[Błędy](#)

Kodowanie

Proces programowania



- Przechowywanie informacji
- Opisanie składni
- Zmienne
- Operator przypisania
- Proste typy danych
- Stałe
- Ciąg operatorów
- Komentarz
- Wyjście
- Wejście
- Szablon Aplikacji
- Case Study
- Kodowanie
 - ❖ Proces programowania
 - ❖ Makefile
- Błędy

[Przechowywanie informacji](#)

[Opisanie składni](#)

[Zmienne](#)

[Operator przypisania](#)

[Proste typy danych](#)

[Stałe](#)

[Ciąg operatorów](#)

[Komentarz](#)

[Wyjście](#)

[Wejście](#)

[Szablon Aplikacji](#)

[Case Study](#)

[Kodowanie](#)

❖ Proces programowania

❖ Makefile

[Błędy](#)

● Kompilator GNU

```
gcc -Wall -g revert4symbols.c -o revert
```

Narzędzie *make*

Przechowywanie informacji

Opisanie składni

Zmienne

Operator przypisania

Proste typy danych

Stałe

Ciąg operatorów

Komentarz

Wyjście

Wejście

Szablon Aplikacji

Case Study

Kodowanie

❖ Proces programowania

❖ **Makefile**

Błędy

- Plik Makefile

```
CFLAGS=-Wall -g
```

```
clean:
```

```
rm -f revert4symbols
```

- ◆ tabulacje!

- ◆ kompilacja:

```
make revert4symbols
```

Narzędzie make

Przechowywanie informacji

Opisanie składni

Zmienne

Operator przypisania

Proste typy danych

Stałe

Ciąg operatorów

Komentarz

Wyjście

Wejście

Szablon Aplikacji

Case Study

Kodowanie

❖ Proces programowania

❖ Makefile

Błędy

● Plik Makefile

```
CFLAGS=-Wall -g
```

```
all:                revert4symbols
```

```
clean:              rm -f revert4symbols
```

◆ kompilacja:

```
make
```

[Przechowywanie informacji](#)

[Opisanie składni](#)

[Zmienne](#)

[Operator przypisania](#)

[Proste typy danych](#)

[Stałe](#)

[Ciąg operatorów](#)

[Komentarz](#)

[Wyjście](#)

[Wejście](#)

[Szablon Aplikacji](#)

[Case Study](#)

[Kodowanie](#)

[Błędy](#)

❖ Błędy

Błędy

Błędy

[Przechowywanie informacji](#)

[Opisanie składni](#)

[Zmienne](#)

[Operator przypisania](#)

[Proste typy danych](#)

[Stałe](#)

[Ciąg operatorów](#)

[Komentarz](#)

[Wyjście](#)

[Wejście](#)

[Szablon Aplikacji](#)

[Case Study](#)

[Kodowanie](#)

[Błędy](#)

❖ Błędy

- Kompilacji
- Uruchomienia
- Logiczne