

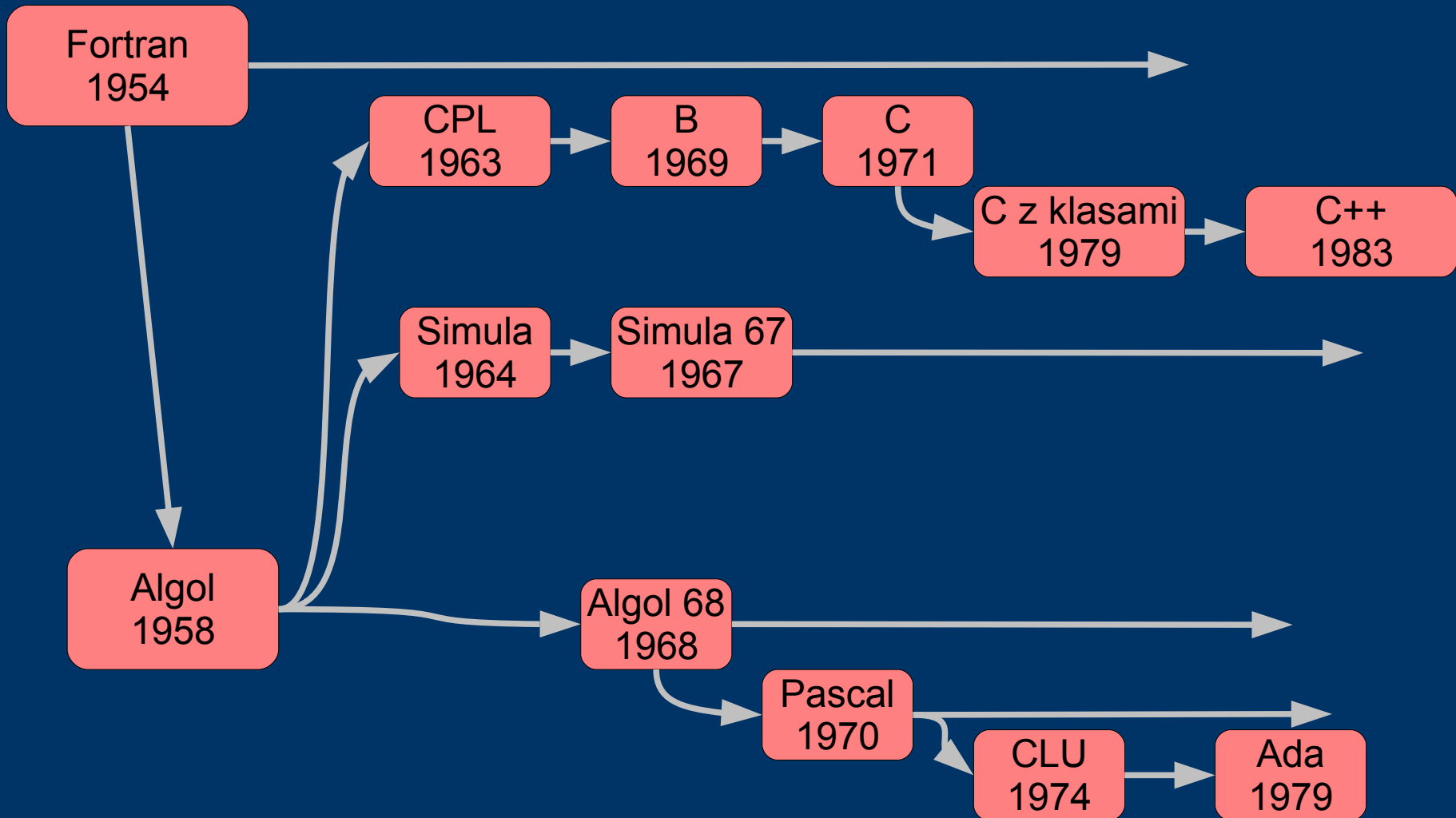
# Język ludzki – kod maszynowy



# Historia języków programowania

1954

2008



# Od kodu do programu

```
if (liczba_pkt > 50)
    zaliczenie=true
else
    zaliczenie=false
```

Kompilator (kompilacja)

```
0011100101010111010000111
001001101001101101010100
001001001000100010000100
```

Linker (konsolidacja)

program.exe  
działająca aplikacja

Napisany przez programistę kod poddawany jest **kompilacji**. Po sprawdzeniu poprawności kodu **kompilator** przekształca go na kod maszynowy. Następnie **linker** łączy z kodem potrzebne do jego działania biblioteki i tworzy **plik wykonywalny** aplikacji. Rozszerzenie .exe jest umowne i charakterystyczne dla systemu Windows.

dodatkowe biblioteki  
i moduły

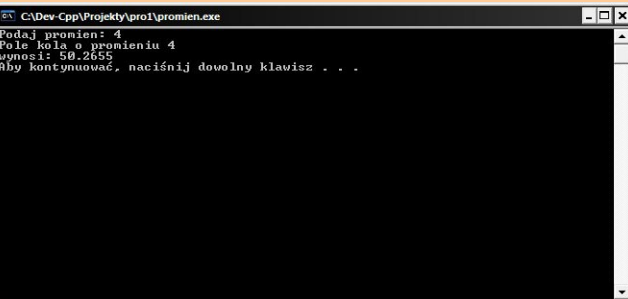
# Jak powstaje program

## ZADANIA PROGRAMU:

1. obliczanie pola
2. obliczanie objętości
3. ...

## ALGORYTM:

1. pobierz promień
2. oblicz pole
3. wyświetl pole
4. ...



```
int main() {  
    int pole;  
    cin >> r;  
    pole = r * r + 3;  
    cout << pole; }  
}
```

START

pobierz r

oblicz pole

wyświetl pole

KONIEC

# Algorytmy

- **Algorytmem** nazywamy skończony ciąg czynności, przekształcający zbiór danych wejściowych na zbiór danych wyjściowych.
  - Algorytmy, które wykonują działania matematyczne na danych liczbowych, nazywamy **algorytmami numerycznymi**.
  - Algorytmy, w których kolejność czynności jest zawsze taka sama i niezależna od wartości danych wejściowych, nazywamy **algorytmami sekwencyjnymi** lub inaczej **linowymi**.
- 
-

# *Etapy konstruowania algorytmu*

1. Sformułowanie zadania.
  2. Określenie danych wejściowych.
  3. Określenie wyniku i sposobu jego prezentacji.
  4. Ustalenie metody wykonania zadania.
  5. Zapisanie algorytmu za pomocą wybranej metody.
  6. Analiza poprawności rozwiązania.
  7. Testowanie rozwiązania dla różnych danych – algorytm powinien być uniwersalny, tzn. działać dla różnych zestawów danych.
  8. Ocena skuteczności algorytmu.
- 
-

# Sposoby zapisu algorytmu

## LISTA KROKÓW:

1. Pobierz dane promienia
2. Oblicz wartość pola
3. Wyświetl wynik na ekranie
4. Zakończ

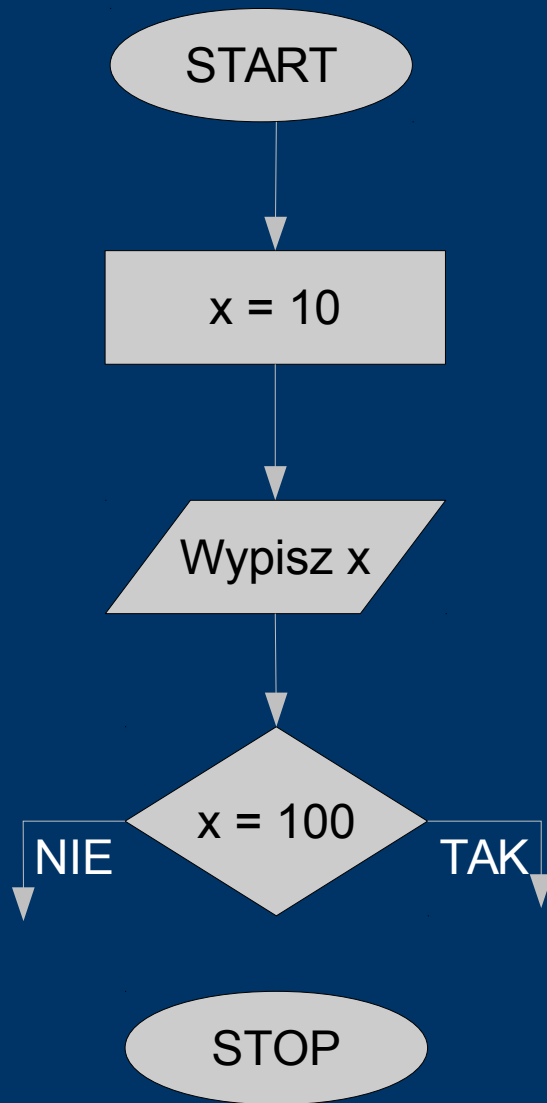
## PSEUDOJĘZYK:

Początek;  
Rzeczywiste  $r$ ;  
 $\text{Pole} = r * r * \text{PI}$ ;  
Wyświetl Pole;  
Koniec.

## SCHEMAT BLOKOWY

- przedstawia algorytm w postaci symboli graficznych.

# Elementy schematu blokowego



**Blok graniczny** – początek algorytmu

**Blok operacyjny** – operacje zmieniające wartości zmiennych.

**Blok wejścia/wyjścia** – operacje wprowadzania i wyprowadzania danych.

**Blok decyzyjny** – wybór jednej z dwóch możliwych dróg działania.

**Blok graniczny** – koniec algorytmu



# Specyfikacja problemu

- **Specyfikacją problemu algorytmicznego** nazywamy dokładny opis zadania, które ma zostać rozwiązane, oraz podanie danych wejściowych i danych wyjściowych wraz z ich typami.
  - **Zmienną** nazywamy obiekt występujący w algorytmie, określony przez nazwę i służący do zapamiętywania pewnych danych.
  - **Typ zmiennej** – określa rodzaj wartości, jakie może ona przechowywać.
  - **Zmienna pomocnicza** – umożliwia realizację algorytmu, służy do przechowywania danych przejściowych.
  - **Stałe** – przechowują wartości, które w trakcie działania algorytmu nie ulegają zmianie, np. wartość  $\pi$  (3,14).
- 
-

# ***Złożoność algorytmu***

- Na **złożoność obliczeniową** algorytmu składają się:
    - **złożoność pamięciowa** – zależy od liczby i rozmiaru danych wykorzystywanych w algorytmie i oznacza rozmiar pamięci potrzebny do realizacji algorytmu;
    - **złożoność czasowa** – określa czas potrzebny do wykonania algorytmu w odniesieniu do rozmiaru danych wejściowych. Za jednostkę czasu przyjmuje się wykonanie **operacji dominującej** (najczęściej wykonywanej) algorytmu (np. dodawanie, mnożenie w a. numerycznych, czy porównanie i przestawianie w a. porównujących).
  - **Pesymistyczna złożoność obliczeniowa** – to złożoność algorytmu dla pesymistycznego zestawu danych.
- 
-

# Podstawowe typy danych

Typ	Typowy rozmiar w bajtach	Zakres
char	1	-128 ... 127
int	2, 4 lub 8	-32 768 ... 32 767
long int	4	-2 147 483 648 ... 2 147 483 647
float	4	6 cyfr znaczących -3,4E38 ... 3,4E38
double	8	10 cyfr znaczących -1,7E308 ... 1,7E308
bool	1	true, false

# Stałe i zmienne

- Stałe tekstowe i liczbowe nie zmieniają swojej wartości w trakcie działania programu.  
Przykładowa definicja:  
`const string KP = "Kot Pulpet";`
  - Zmienne przed pierwszym użyciem zawsze należy zadeklarować określając typ danych przez nią przechowywanych. Dobrze jest również inicjować zmienne, czyli nadawać im początkowe wartości.
  - Deklaracja zmiennej: `int wynik;`  
Definicja zmiennej: `int wynik = 0;`  
Inicjacja zmiennej: `wynik = 0;`
- 
-

# *Nazywanie stałych i zmiennych*

- Nazwa powinna określać przeznaczenie stałej lub zmiennej. Może zawierać znaki alfabetu (poza narodowymi), cyfry i podkreślenia, ale nie może rozpoczynać się od cyfry. W nazwach stałych używa się zwyczajowo DUŻYCH liter.
  - Nie można używać słów zarezerwowanych w języku C:  
auto, break, case, char, class, const, continue,  
default, do, double, else, enum, extern, float, for,  
goto, if, int, long, private, register, return, short,  
signed, sizeof, static, struct, switch, try, typedef,  
unsigned, void, while.
- 
-

# Operatory i wyrażenia

- Wyrażenie jest kombinacją stałych, zmiennych i operatorów, stosowaną do zapisu operacji matematycznych lub innych.

Symbol	Znaczenie	Przykład
+	dodawanie	<code>x=a+b;</code>
-	odejmowanie	<code>x=a-b;</code>
*	mnożenie	<code>x=a*b;</code>
/	dzielenie	<code>x=a/b;</code>
%	reszta z dzielenia	<code>x=a%b;</code>
++	inkrementacja o 1	<code>x++; ++x;</code>
--	dekrementacja o 1	<code>x--; --x;</code>
=	przypisanie	<code>x=2*3; x=y=4;</code>
,	łączenie	<code>int x,y,z;</code>

# Operatory skróconego przypisania

Operator	Nazwa	Przykład	Znaczenie
<b>+=</b>	przypisanie z dodawaniem	<code>x+=2;</code>	<code>x=x+2;</code>
<b>-=</b>	przypisanie z odejmowaniem	<code>x-=2;</code>	<code>x=x-2;</code>
<b>*=</b>	przypisanie z mnożeniem	<code>x*=2;</code>	<code>x=x*2;</code>
<b>/=</b>	przypisanie z dzieleniem	<code>x/=2;</code>	<code>x=x/2;</code>
<b>%=</b>	przypisanie reszty z dzielenia	<code>x%=2;</code>	<code>x=x%7;</code>

# Operatory relacji

Operator	Nazwa	Przykład	Wynik
<b>==</b>	równe	12==(10+2);	1 (true)
<b>!=</b>	różne od	7!=(9-2);	0 (false)
<b>&lt;</b>	mniejsze	3<9;	1 (prawda)
<b>&gt;</b>	wieksze	4>(2+7);	0 (fałsz)
<b>&lt;=</b>	mniejsze lub równe	5<=(2+3)	1 (prawda)
<b>&gt;=</b>	wieksze lub równe	4>=5	0 (fałsz)



# Zadanie

- Skonstruuj algorytm obliczający pole trójkąta o bokach  $a$ ,  $b$  i  $c$  podanych na wejściu. Uwzględnij błędne wartości danych wejściowych.
- Wykorzystaj wzór Herona:  
$$\text{pole} = \sqrt{p(p-a)(p-b)(p-c)}$$
  
gdzie  $p = \frac{(a+b+c)}{2}$
- oraz zależność mówiącą o tym, że *suma dowolnych dwóch boków trójkąta jest większa od długości trzeciego boku.*

# *Przykładowy algorytm*

- Zadeklaruj zmienne  $a$ ,  $b$  i  $c$ .
  - Pobierz od użytkownika wartości  $a$ ,  $b$  i  $c$  (długość boków trójkąta).
  - Sprawdź, czy podane boki tworzą trójkąt.
  - Jeżeli tak, oblicz współczynnik  $p$ .
  - Oblicz pole trójkąta przy wykorzystaniu wzoru Herona.
  - Wyprowadź wynik na ekranie.
- 
-