

# Podstawy programowania (1)

**doc. dr inż. Tadeusz Jeleniewski**

**Konsultacje – pokój 19**

**Poniedziałki, godz. 9:45 – 11:20**

***e-mail: [tadeusz.jeleniewski@neostrada.pl](mailto:tadeusz.jeleniewski@neostrada.pl)***

# Podstawy programowania (1) - wykład

1. **Wprowadzenie** Algorytmiczne języki programowania. Struktura programu źródłowego w języku C++. Proces kompilacji, konsolidacji i uruchamiania. Przykład prostego programu. Jednostki składniowe języka.
2. **Podstawowe pojęcia** Typy wartości, zmienne. Reprezentacja danych w komputerze. Typy całkowite i rzeczywiste. Inicjacja zmiennych.
3. **Komunikacja programu z otoczeniem** Funkcje *scanf*, *printf*, *gets*, *puts*. Podejście "obiektowe" - strumienie standardowe *cin*, *cout*
4. **Wskaźniki (1)** Zmienne i ich adresy. Odwoływanie się do zmiennych przez nazwę lub adres. Zmienne wskaźnikowe - notacja, znaczenie. Arytmetyka wskaźników.
5. **Operatory i wyrażenia** Przypisanie proste i arytmetyczne, wyrażenia arytmetyczne, inkrementacja i dekrementacja, operatory logiczne, relacje, operatory bitowe, wyrażenie warunkowe
6. **Sterowanie wykonaniem programu** Instrukcja *if*, *if ... else*, *else if*. Zagnieżdżanie instrukcji *if*. Instrukcja *switch ... case*

- 7. Instrukcje iteracyjne** Pojęcie pętli programowej. Pętla for, działanie i zastosowanie. Pętle z badaniem warunku na początku (while) i na końcu (do ... while). Zagnieżdżanie pętli. Instrukcje break oraz continue
- 8. Funkcje** Pojęcie funkcji w C++. Deklaracja (prototyp) i definicja funkcji. Zwracanie wartości funkcji. Przekazywanie parametrów przez wartość, adres i referencję
- 9. Tablice i łańcuchy** Typ strukturalny - tablica, deklaracja i inicjacja wartości elementów. Tablice wielowymiarowe jako tablice tablic. Tablice jako argumenty funkcji. Łańcuch znaków i jego reprezentacja w postaci tablicy znakowej. Funkcje przetwarzające łańcuchy.
- 10. Wskaźniki (2)** Związek pomiędzy tablicami a wskaźnikami. Łańcuchy znaków a wskaźniki. Wskaźniki do elementów tablic
- 11. Struktury** Pojęcie struktury, deklaracja struktury, atrybuty dostępu, możliwość definiowania metod. Zagnieżdżanie struktur. Wskaźniki na struktury. Tablice struktur. Unie i ich zastosowanie.
- 12 Obsługa plików** Standardowe plikowe wejście i wyjście. Wejście/wyjście znakowe, łańcuchowe, formatowane, blokowe. Pliki standardowe. Drukarka jako plik wyjściowy

**Przedmiot kończy się zaliczaniem na stopień**

**Kolokwium zaliczeniowe odbędzie się w terminie przedostatniego wykładu w semestrze.**

**Możliwe są 10-minutowe sprawdziany przed rozpoczęciem wykładu.**

## Literatura

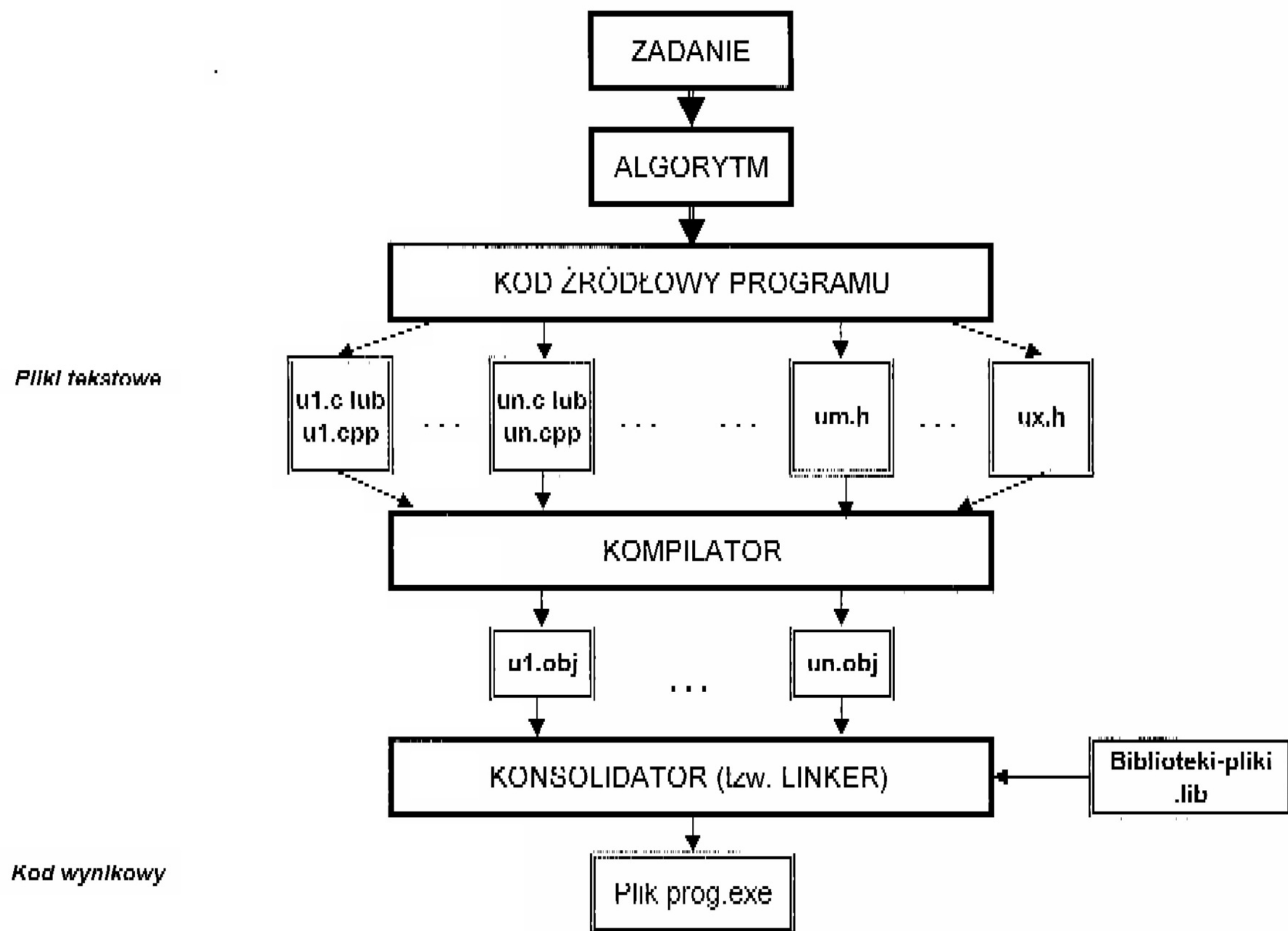
- Robert Lafore *Programowanie w języku C przy użyciu Turbo C++*. Intersoftland, 1995
- Andrzej Zalewski *Programowanie w językach C i C++ z wykorzystaniem pakietu Borland C++*. Wydawnictwo Nakom, Poznań, 1994
- Walter Savitch *Programowanie w tonacji C++*. Wydawnictwo RM, Warszawa, 2005
- Jerzy Grębosz *Symfonia C++*. Oficyna Kallimach, Kraków, 1996
- Herbert Schildt *Informator o języku programowania Borland C++*. Wydawnictwo Nakom, Poznań, 1998
- Jesse Liberty *C++ dla każdego*. Wydawnictwo HELION, 2002

# Wykład 1

- **Co to jest program komputerowy i jak powstaje**
- **Struktura programu źródłowego w języku C++**
- **Jednostki składniowe**

W języku C++ tworzenie programu odbywa się w dwóch etapach: opracowanie **kodu źródłowego** i generowanie **kodu wynikowego**. Przed rozpoczęciem opracowywania kodu źródłowego należy obmyślić **algorytm** rozwiązania postawionego zadania. Opracowanie **kodu źródłowego** czyli tzw **kodowanie** sprowadza się do zapisu wcześniej obmyślonego algorytmu w języku programowania - w naszym przypadku jest nim język C++. Etap generowania kodu wynikowego jest realizowany przez komputer. Specjalny program - **kompilator** analizuje kod źródłowy programu pod względem poprawności leksykalnej, składniowej i częściowo semantycznej (znaczeniowej) i po ewentualnym usunięciu przez programistę wykrytych błędów tłumaczy na postać pośrednią, do której inny program (**program łączący - linker**) dołącza brakujące elementy umieszczone w **bibliotekach funkcji**. Po pomyślnym zakończeniu procesu łączenia powstaje **kod wynikowy** - program gotowy do wykonania przez maszynę. Jest to algorytm rozwiązania zadania zapisany w **języku wewnętrznym** komputera. Algorytm nie jest na ogół pozbawiony **błędów wykonania**. Są to błędy, których nie wykrywa kompilator w fazie analizy poprawności programu źródłowego.

Inna kategoria błędów, to błędy wykrywane przez program tłumaczący – kompilator. Są to tzw. ***błędy kompilacji***. Porównać je można do błędów ortograficznych, składniowych oraz gramatycznych, które często zdarza nam się popełniać w języku naturalnym (nawet ojczystym). Rysunek 1 przedstawia schematycznie przebieg procesu tworzenia kodu wynikowego pewnego programu.



Charakterystyczną cechą języka C++ jest możliwość budowy programu z wielu *modułów*.

Modułem w języku C++ może być każdy zbiór zawierający poprawny kod źródłowy.

Program w języku C++ buduje się z *funkcji*. Każda funkcja może (ale nie musi) posiadać *parametry* i określony *typ wartości*.

Aby możliwe było wygenerowanie kodu wynikowego programu w postaci przyjmowanej przez system operacyjny (w systemach operacyjnych DOS, Windows jest to zbiór z rozszerzeniem .exe) w jednym (i tylko w jednym) module programu musi znaleźć się funkcja o nazwie *main*. Od funkcji tej system rozpoczyna wykonywanie programu.

Moduł, który zawiera tą funkcję nazywa się *modułem głównym*.

Najprostszy, poprawny program w C++ może mieć postać:

```
void main(void)
{
}
```

Program ten składa się z bezparametrowej funkcji *main*. Funkcja ta nie zwraca do systemu operacyjnego żadnej wartości. Wewnątrz nawiasów klamrowych znajduje się *blok*, który zawiera *definicję funkcji*.

Przydatność praktyczna takiego programu jest oczywiście znikoma. Wewnątrz bloku stanowiącego definicję funkcji nie umieszczono żadnego polecenia – ten program niczego nie wykonuje.

**Inny, prosty program może mieć postać:**

```
/* Program 1.2 - przykład prostego programu
   napisanego w języku C++.
   Program wyświetla na ekranie komunikat
   powitalny
*/
#include <iostream.h>
void main(void)
{
    cout<<"\nWitaj w świecie C++";
}
```

W tekście tego programu można wyróżnić następujące elementy (*jednostki składniowe*):

- *komentarz* - fragment tekstu pomijany przez kompilator w procesie analizy i tłumaczenia programu źródłowego; *komentarzem* jest dowolny tekst pomiędzy parą symboli dwuznakowych */\** (ukośnik gwiazdka) oraz *\*/* (gwiazdka ukośnik),

*dyrektywa preprocesora* - słowo *include* wraz z poprzedzającym je znakiem *#*; jest to polecenie dla kompilatora, aby przed rozpoczęciem analizy tekstu źródłowego dołączył do niego tzw. *plik nagłówkowy* o nazwie *iostream.h*; w *plikach nagłówkowych* znajdują się definicje *symboli* użytych w programie oraz tzw. *prototypy funkcji*;

- **deklaracja funkcji *main*** - przykładowy program składa się z bezparametrowej funkcji głównej nie reprezentującej sobą żadnej wartości, na co wskazuje słowo **void** poprzedzające nazwę funkcji głównej;
- wewnątrz nawiasów klamrowych znajduje się **blok** zawierający **definicję** funkcji *main*;

**cout** jest **nazwą obiektu** zdefiniowanego w pliku nagłówkowym **iostream.h**; obiekt ten to tzw. **strumień wyjściowy**, do którego za pomocą operatora **<<** (również zdefiniowanego w pliku **iostream.h**) program prześle tekst komunikatu do wyświetlenia na ekranie monitora.

**Przykładowy program wyświetli na ekranie monitora napis (*komunikat*) treści:**

**Witaj w świecie C++  
oraz zakończy działanie.**

**Takie samo zadanie wykona program:**

```
/* Program w1_2 - inna wersja (proceduralna)
   prostego programu, który wyświetla
   komunikat powitalny
*/
#include <stdio.h>
void main(void)
{
    printf("\nWitaj w świecie C++");
}
```

Do wyświetlenia komunikatu użyto tutaj *funkcji standardowej* o nazwie *printf*, której parametrem jest *łańcuch znaków*.

Funkcja ta nie zwraca żadnej wartości. Warto zauważyć, że w tym przypadku do tekstu programu dołączono inny niż poprzednio plik nagłówkowy. W pliku `stdio.h` znajduje się *prototyp funkcji standardowej* użytej w naszym przykładzie. Definicja tej funkcji znajduje się w *bibliotece funkcji standardowych*, która jest automatycznie przeglądana przez program łączący w fazie tworzenia *kodu wynikowego*.

***Jednostkami składniowymi*** języka C++ są: ***identyfikatory, słowa kluczowe, znaki przestankowe, stałe, literały łańcuchowe, operatory.***

Identyfikatorem w języku C++ może być ciąg dużych i małych liter, cyfr i znaku podkreślenia **`_`**.

Identyfikator jest nazwą elementu języka (zmiennej, funkcji itp.). Pierwszy znak identyfikatora musi być literą lub znakiem podkreślenia.

Kompilator C++ rozróżnia pierwsze 32 litery nazwy. Nazwy dłuższe są poprawne. Kompilator rozróżnia duże i małe litery nazwy.

Nazwa **`Yuma_15_10`** jest identyfikatorem zupełnie innego elementu programu niż **`yUMA_15_10`**

**Słowa kluczowe** są to identyfikatory zastrzeżone dla specjalnych celów. Mogą być wykorzystywane tylko zgodnie z ich przeznaczeniem.

Zestawienie wszystkich słów kluczowych można znaleźć np. w książce A.Zalewskiego (str.69-70).

Kilka przykładów słów kluczowych języka C/C++:

`void`    `for`    `int`    `short`    `float`    `return`    `do`  
`while`    `struct`    `class`    `continue`    `switch`  
`case`

Słowa kluczowe to instrukcje języka, nazwy typów, dyrektywy preprocesora i kompilatora itp.

**Znakami przestankowymi są:**

**[ ] ( ) { } , ; : ... \* = #**

·nawiasy kwadratowe [ ] - definiowanie tablic i wskazywanie ich elementów,

·nawiasy okrągłe ( ) - grupowanie wyrażeń, wywołania funkcji, listy parametrów,

·nawiasy klamrowe { } - początek i koniec instrukcji złożonej,

·przecinek , - oddzielanie elementów listy,

·średnik ; - zakończenie instrukcji,

·dwukropek : - poprzedzający go ciąg znaków jest etykietą,

·wielokropek ... - deklarowanie funkcji o zmiennej liczbie parametrów,

·gwiazdka \* - deklarowanie zmiennych wskaźnikowych,

·znak równości = - oddzielanie deklaracji zmiennej od części inicjującej tą zmienną,

·znak # - oznaczenie dyrektywy preprocesora.