Programowanie strukturalne i obiektowe

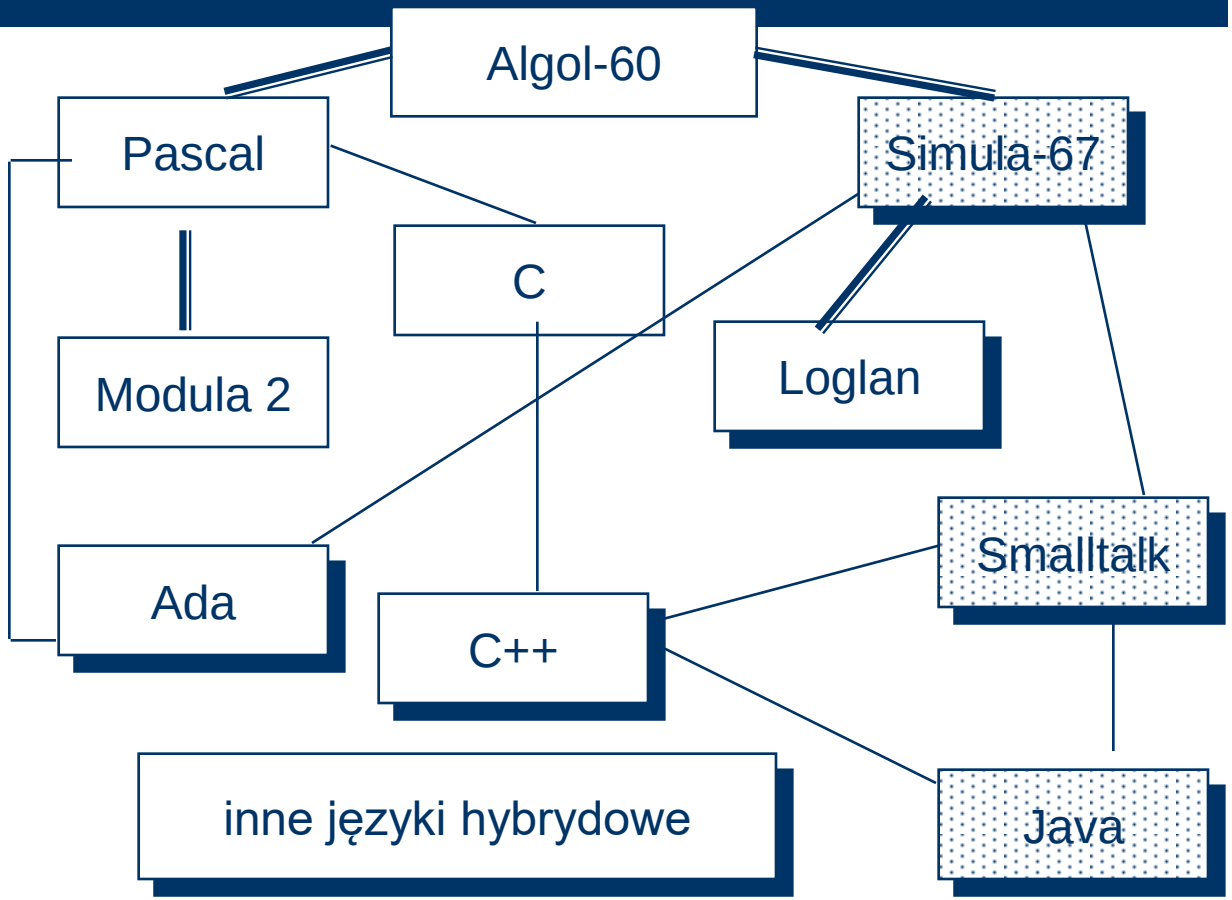
Język C++

Maciej Dawid

Literatura

- **Bjarne Stroustrup – „Język C++”**
- **Jerzy Grębosz – „Symfonia C++”**
- Bruce Eckel – „Thinking in C++. Edycja polska”
- Stanley B. Lippman – „Podstawy języka C++”

Historia


Algol-60
Pascal
Simula-67
C
Modula 2
Loglan
Ada
Smalltalk
C++
inne języki hybrydowe
Java

Historia

- Simula-67: lata 60-te
- Smalltalk: lata 70-te, rozwój w latach 90-tych
- Pascal: początek lat 70-tych
- C: lata 70/80
- Ada: początek lat 80-tych
- C++: początek lat 80-tych
- Java: połowa lat 90-tych

- popularność programowania obiektowego: lata 90

Historia C++

- Lata 70 powstaje język C
- Twórcy C - Brian Kernighan i Dennis Ritchie
- 1979 pojawia się nowy język zbliżony do C
- Pierwsza nazwa języka opartego na C nosi nazwę „C with classes”
- 1983 nowy język zostaje nazwany C++
- Twórcą języka jest zespół programistów pod kierownictwem Bjarne Stroustrupa
- 1989 roku powstał pierwszy kompilator języka C++

Cechy zaczerpnięte z innych języków

- C – zaczerpnięta składnia
- Simula67 - zaczerpnięto funkcje wirtualne oraz dziedziczenie
- Algol68 - był "dawcą" przeciążonych operatorów
- Ada i Clu - były inspiracją dla wprowadzenia szablonów

Zalety języka C++

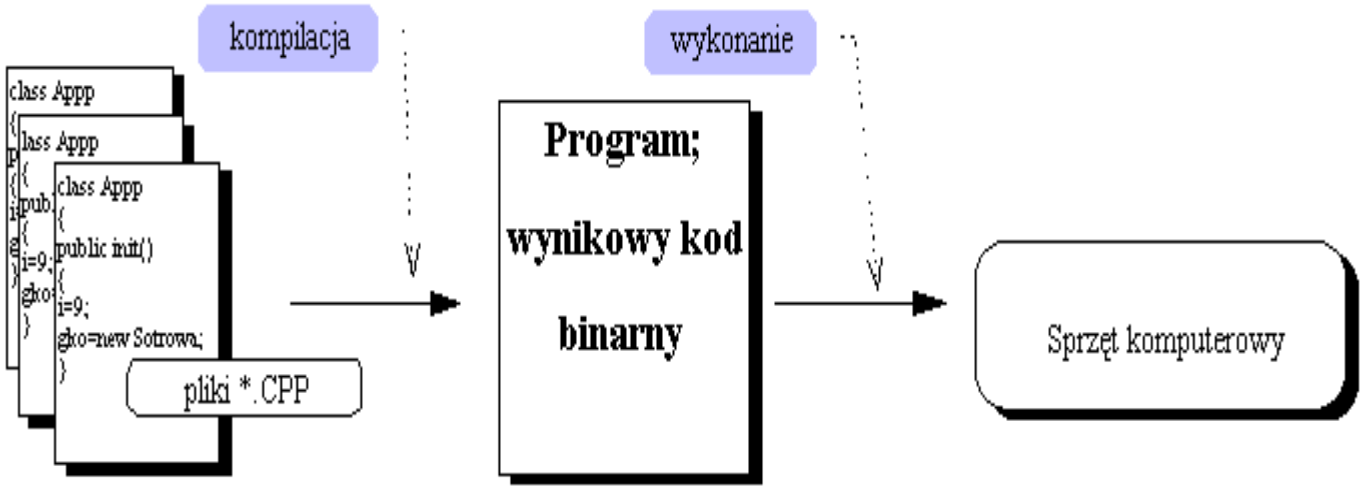
- nowoczesność
- duże możliwości
- wspólne cechy z innymi językami programowania
- uniwersalność
- szybkość

Kompilatory i środowiska

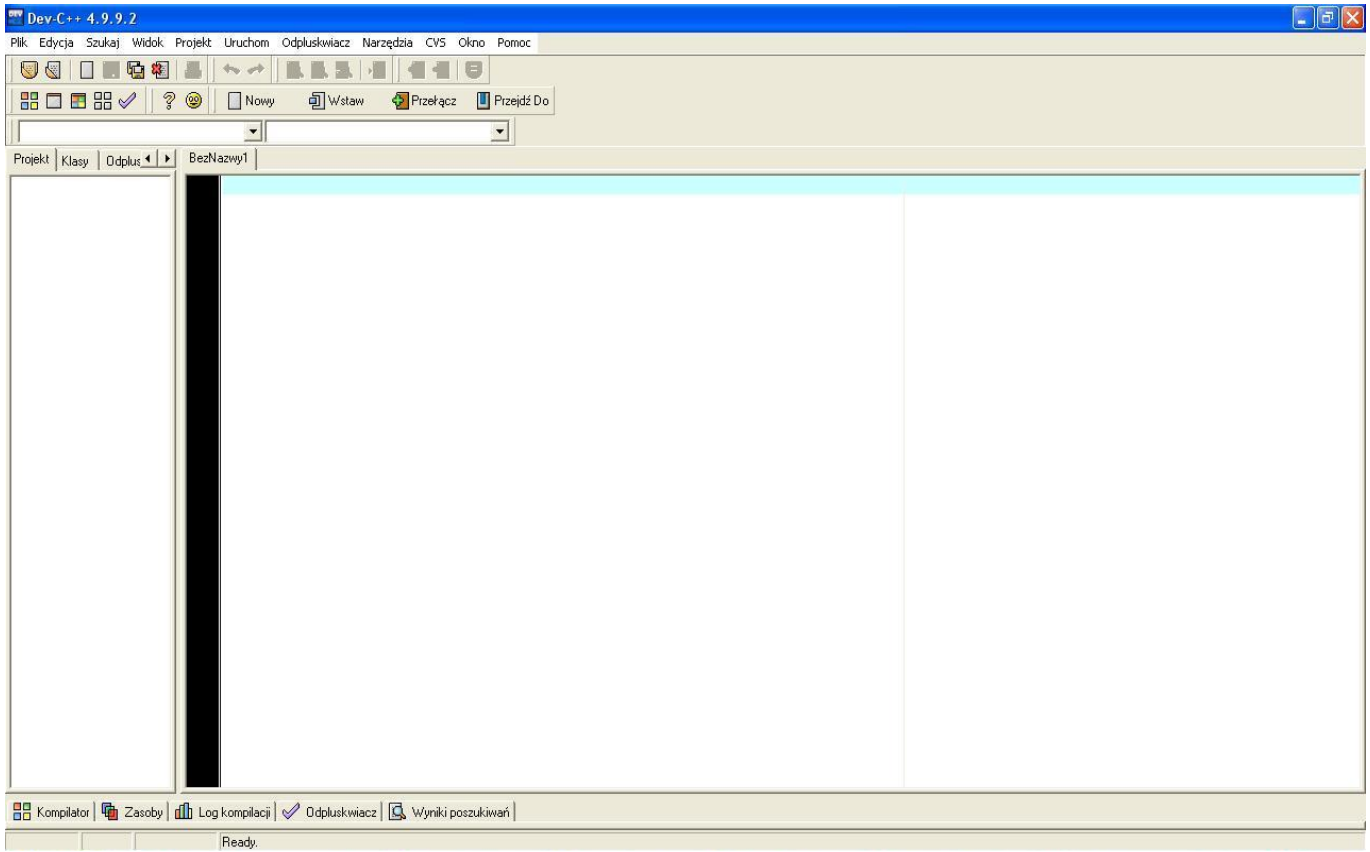
http://pl.cpp.wikia.com/wiki/Kompilatory

- DJGPP
- Microsoft Visual C++
- C++ Builder
- KDevelop
- Intel C++ Compiler
- GCC – GNU Compiler Colection (g++)
- Dev-C++ (http://www.bloodshed.net/devcpp.html)

Kompilacja w C++



Dev-C++


Dev-C++ 4.9.9.2
Plik Edycja Szukaj Widok Projekt Uruchom Odpluskwiacz Narzędzia CVS Okno Pomoc
Nowy
Wstaw
Przełącz
Przejdź Do
Projekt
Klasy
BezNazwy1
Kompilator
Zasoby
Log kompilacji
Odpluskwiacz
Wyniki poszukiwań
Ready.

Pierwszy program

```
#include <iostream>
using namespace std;

int main ( )
{
    cout<<"Lubie programowanie !"<<endl;
    system("PAUSE");
    return 0;
}
```

#include <iostream>

- #include – dyrektywa dołączania pliku
- iostream – biblioteka strumienia wejścia i wyjścia
- iostream.h
- Funkcje biblioteczne to najzwyklejsze funkcje dostarczone przez producenta kompilatora. Są one skompilowane i gotowe do pracy.
- cstring, cstdlib, cmath, …

using namespace std;

```
#include <iostream.h>

int main ( )
{
    std::cout<<"Lubie programowanie !"<<std::endl;
    system("PAUSE");
    return 0;
}
```

int main ( )

- int – typ integer (liczba całkowita)
- main () – funkcja główna programu, musi występować w programie
- void main ()

```
main ()
{
 zawartość funkcji głównej
}
```

cout<<"Lubie programowanie !"<<endl;

- cout – strumień wyjścia na ekran
- << - operator przekierowania na strumień wyjścia
- " " – tekst
- endl – przejście do nowej linii
- cout<<"Lubie programowanie ! \n";
- \znak specjalny
- ; seperator: kończy ciąg instrukcji
- printf ("Lubie programowanie\n")

system("PAUSE");

- system () – wywołanie funkcji systemowej
- PAUSE – funkcja pauzująca (tylko systemy operacyjne Microsoftu)
- getchar (); - funkcja czekająca na podanie znaku z klawiatury

return 0;

- Zwraca wartość funkcji głównej
- Wymagane przez specyfikacje języka

Skrócona wersja

```
#include <iostream>
using namespace std;
main(){cout<<"Lubie programowanie!\n";}
```

Komentarze

- Nie mają żadnego wpływu na działanie programu
- // komentarz jednolinijkowy
- od sekwencji // do końca linii
- /* komentarz o dowolnej ilości linii */
- Komentarz nie jest brany pod uwagę przez kompilator

Komentarze

```
#include <iostream>
using namespace std;

int main ( )  // funkcja główna
{
    /* cout<<"Lubie programowanie !"<<endl;
    system("PAUSE");
    return 0; */
}
```

Słowa kluczowe

- Słowa kluczowe - są integralną częścią składni języka. Oznacza to, że są one zarezerwowane i nie mogą być używane w innych celach niż to pierwotnie założono, nie można ich np. wykorzystać do nadawania nazw zmiennych czy funkcji.

Przykładowe słowa kluczowe

- asm, break, case, catch, char, class, const, continue, default, delete, do, double, else, enum, extern, float, for, friend, goto, if, inline, int, long, new, operator, private, protected, public, register, return, short, signed, sizeof, static, struct, switch, this, throw, try, typedef, unsigned, virtual, void, volatile, while

C++

Zmienne i typy

Zmienne

- Pod pojęciem zmiennej rozumiemy zwykle komórkę pamięci, której rozmiar zależy od typu danej, którą chcemy w niej przechowywać. Inny będzie więc rozmiar zmiennej dla liczby całkowitej, a inny dla ciągu znaków.

- Każda zmienna posiada swój adres w pamięci, jednak odwoływanie się do zmiennej poprzez adres byłoby dość kłopotliwe, dlatego też zmiennym można przyporządkować nazwy. Nazwa składa się z ciągu liter i cyfr, nie może jednak zawierać „polskich liter”, spacji ani rozpoczynać się od cyfry. Nie może być oczywiście taka sama jak istniejące słowo kluczowe. W języku C++ rozróżniane są duże i małe litery.

Typy danych

Typ danych	Liczba bitów	Zakres wartości
char	8	-128:127 lub jeden znak ASCII
int	32	-2 147 483 648 : 2 147 483 647
unsigned int	32	0 : 4 294 967 295
short int	16	-32 768 : 32 767
float	32	$3,4*10^{-38}$: $3,4*10^{38}$
double	64	$2,2*10^{-308}$: $1,8*10^{308}$
bool	8	**True** lub **False**

Deklaracje zmiennych

- Każda nazwa w C++ zanim zostanie użyta, musi zostać zadeklarowana.
- *Deklaracja* informuje kompilator, że dana nazwa reprezentuje obiekt określonego typu, ale nie rezerwuje dla niego miejsca w pamięci.
- *Definicja* zaś - dodatkowo rezerwuje miejsce. Definicja jest miejscem, gdzie powołuje się obiekt do życia.
- Oczywiście, *definicja* jest zawsze również *deklaracją*. Deklarować obiekt w programie można wielokrotnie, natomiast definiować można tylko raz.
- Zawsze trzeba zdefiniować zmienną przed jej pierwszym użyciem.
- Przykład:

```
int licznik;                // definicja + deklaracja
extern int licznik;         // deklaracja ( tylko!)
```

Przykłady definicji zmiennych

```
TypZmiennej   NazwaZmiennej;
int a;
int b=7;
float c=3.14;
double d;
d=2.543;
char e='X';
int f,zmienna,z;
```

Przykładowy program

```
#include <iostream>
using namespace std;

int main()
{
    int a,b;
    int wynik;
    a=5;
    b=10;
    wynik=a+b;
    cout<<"Wynik dodawania wynosi"<<wynik<<endl;
    system ("PAUSE");
    return 0;
}
```

Standardowy strumień wejścia

```
#include <iostream>
using namespace std;
int main()
{
    int a,b;
    cout<<"Podaj a"<<endl;
    cin>>a;
    cout<<"Podaj b"<<endl;
    cin>>b;
    cout<<"Wynik dodawania wynosi"<<a+b<<endl;
    system ("PAUSE");
    return 0;
}
```

Znaki sterujące i specjalne

Znaki sterujące

\b - cofacz (backspace)

\f - nowa strona

\n - nowa linia

\t - tabulator

\v - tabulator pionowy

\a - sygnał dźwiękowy

Znaki specjalne

\\ - ukośnik lewostronny

\’ - apostrof

\” - cudzysłów

\0 - NULL, znak o kodzie 0

\? - znak zapytania

Przykład

```
#include <iostream>
using namespace std;
int main()
{
    int a,b;
    cout<<"\t Podaj a \n";
    cin>>a;
    cout<<"\t Podaj b \n";
    cin>>b;
    cout<<"\a Wynik dodawania wynosi \v"<<a+b<<endl;
    system ("PAUSE");
    return 0;
}
```

C++

Wyrażenia i operatory

Wyrażenie

- W języku C++ każde działanie jest związane z pewnym wyrażeniem.
- Termin **wyrażenie** oznacza sekwencję operatorów i operandów (argumentów), która określa **operacje**, tj. rodzaj i kolejność obliczeń.
- **Operandem** nazywa się wielkość, poddaną operacji, która jest reprezentowana przez odpowiedni **operator**. *Np.* $a+b$
- Operatory, które oddziaływują tylko na jeden operand, nazywa się **jednoargumentowymi** (unarnymi).
- Operatory dwuargumentowe nazywa się **binarnymi**; ich argumenty określa się jako **operand lewy** i **operand prawy**. Niektóre operatory reprezentują zarówno operacje jednoargumentowe, jak i dwuargumentowe.

Operatory arytmetyczne

Symbol operatora	Funkcja	Zastosowanie
+	Dodawanie	wyrażenie+wyrażenie
-	Odejmowanie	wyrażenie-wyrażenie
*	Mnożenie	wyrażenie*wyrażenie
/	Dzielenie	wyrażenie/wyrażenie
%	Reszta z dzielenia	wyrażenie%wyrażenie

Operatory relacji

Wszystkie operatory relacji są dwuargumentowe. Jeżeli relacja jest prawdziwa, to jej wartością jest 1; w przypadku przeciwnym wartością relacji jest 0.

Operatory relacji

Symbol operatora	Funkcja	Zastosowanie
<	Mniejszy	wyrażenie<wyrażenie
<=	Mniejszy lub równy	wyrażenie<=wyrażenie
>	Większy	wyrażenie>wyrażenie
>=	Większy lub równy	wyrażenie>=wyrażenie
==	Równy	wyrażenie==wyrażenie
!=	Nierówny (różny od)	wyrażenie!=wyrażenie

Operatory logiczne

Wyrażenia połączone dwuargumentowymi operatorami logicznymi koniunkcji i alternatywy są zawsze wartościowane od strony lewej do prawej. Dla operatora **&&** otrzymujemy wartość 1 (prawda) wtedy i tylko wtedy, gdy wartościowanie obydwu operandów daje 1. Dla operatora || otrzymujemy wartość 1, gdy co najmniej jeden z operandów ma wartość 1.

Operatory logiczne

Symbol operatora	Funkcja	Zastosowanie
!	Negacja	!wyrażenie
&&	Koniunkcja	wyrażenie&&wyrażenie
||	Alternatywa	wyrażenie||wyrażenie

Operatory bitowe

Język C++ oferuje sześć tzw. **bitowych operatorów logicznych**, które interpretują operandy jako uporządkowany ciąg bitów. Każdy bit może przyjmować wartość 1 lub 0.

Operatory bitowe

Symbol operatora	Funkcja	Zastosowanie
&	Bitowa koniunkcja	wyrażenie&wyrażenie
|	Bitowa alternatywa	wyrażenie|wyrażenie
^	Bitowa różnica symetryczna	wyrażenie>wyrażenie
<<	Przesunięcie w lewo	wyrażenie<<wyrażenie
>>	Przesunięcie w prawo	wyrażenie>>wyrażenie
~	Bitowa negacja	~wyrażenie

Operatory przypisania

Symbol operatora	Zapis skrócony	Zapis rozwinięty
+=	a+=b	a=a+b
-=	a-=b	a=a-b
*=	a*=b	a=a*b
/=	a/=b	a=a/b
%=	a%=b	a=a%b
>>=	a>>=b	a=a>>b
<<=	a<<=b	a=a<<b
&=	a&=b	a=a&b
!=	a|=b	a=a|b
^=	a^=b	a=a^b

Operatory inkrementacji i dekrementacji

- W języku C++ istnieją operatory, służące do zwięzłego zapisu zwiększania o 1 (++) i zmniejszania o 1 (--) wartości zmiennej. Zamiast zapisu
  - `n=n+1` (lub `n+=1`)
  - `n=n-1` (lub `n-=1`)
- krótko
  - `++n`, `n++`
  - `--n`, `n--`
- przy czym nie jest obojętne, czy dwuznakowy operator "++" lub "--" zapiszemy przed, bądź za nazwą zmiennej. Notacja przedrostkowa (++n) oznacza, że wyrażenie ++n zwiększa n zanim wartość n zostanie użyta, natomiast n++ zwiększa n po użyciu dotychczasowej wartości n. Tak więc wyrażenia ++n oraz n++ (i odpowiednio --n oraz n--) są różne.