Języki i paradygmaty programowania 1 – studia stacjonarne 2025/26

Lab 2. Zintegrowane środowisko programistyczne Visual Studio – kontynuacja, wyrażenia arytmetyczne, relacyjne, logiczne. Priorytety operatorów.

1. Słowa kluczowe języka C.
   https://learn.microsoft.com/pl-pl/cpp/c-language/c-keywords?view=msvc-170#standard-c-keywords
2. Instrukcje języka C.
   https://learn.microsoft.com/pl-pl/cpp/c-language/statements-c?view=msvc-170
3. Limity wielkości typów danych

```c
#include <stdio.h>
#include <limits.h> //integer
#include <float.h>  //floating point

int main(int argc, char* argv[])
{
	printf("integer range:\t%d\t%d\n", INT_MIN, INT_MAX);
	printf("long int range:\t%ld\t%ld\n", LONG_MIN, LONG_MAX);
	printf("float range:\t%e\t%e\n", FLT_MIN, FLT_MAX);
	printf("double range:\t%e\t%e\n", DBL_MIN, DBL_MAX);
	printf("long double range:\t%e\t%e\n", LDBL_MIN, LDBL_MAX);
	printf("float-double epsilon:\t%e\t%e\n", FLT_EPSILON, DBL_EPSILON);
	return 0;
}
```

Epsilon maszynowy (`FLT_EPSILON`, `DBL_EPSILON`) – wartość określająca precyzję obliczeń numerycznych wykonywanych na liczbach zmiennoprzecinkowych.
Jest to największa liczba nieujemna, której dodanie do jedności daje wynik równy 1. Innymi słowy, 1 + ε = 1 i żadna liczba większa od ε nie spełnia już tego warunku.

4. Tabela operatorów

Lp.	Operatory			łączność	przykład
1.	()	[]	-> .	lewostronna	fun(), tab[], s.a, s->n
2.	- (typ) ! & *	++ -- sizeof	~	prawostronna	-a, (int) a, &r, *p, ++i
3.	* / %			lewostronna	a * b, 3 / c
4.	+ -			lewostronna	a-b
5.			<< >>	lewostronna	
6.		< <= > >=		lewostronna	a<=b
7.		== !=		lewostronna	a != b
8.			&	lewostronna	
9.			^	lewostronna	
10.			|	lewostronna	
11.		&&		lewostronna	war1 && war2
12.		||		lewostronna	war1 || war2
13.	?:			prawostronna	wyr1?wyr2:wyr3
14.	=	+= -= itd.		prawostronna	x += 1
15.	,			lewostronna	wyr1, wyr2, wyr3

5. Bitowe operacje logiczne.

Bitowe operatory logiczne pozwalają na manipulowanie bitami. Mogą działać jedynie dla zmiennych stałopozycyjnych.

~	negacja bitowa (NOT)(uzupełnienie jedynkowe),
<< >>	przesunięcie bitowe w lewo, w prawo,
&	koniunkcja bitowa (AND),
|	alternatywa bitowa (OR)
^	alternatywa rozłączna (XOR) (bitowa różnica symetryczna).

```
 "~" | a      "&" | a | b      "|" | a | b      "^" | a | b
-----+---    -----+---+---    -----+---+---    -----+---+---
  0  | 1       0  | 0 | 0       0  | 0 | 0       0  | 0 | 0
  1  | 0       1  | 1 | 1       1  | 1 | 1       0  | 1 | 1
             0  | 0 | 1       1  | 0 | 1       1  | 0 | 1
             0  | 1 | 0       1  | 1 | 0       1  | 1 | 0
```

- *negacja bitowa daje w wyniku liczbę, która ma bity równe jeden tylko na tych pozycjach, na których argument miał bity równe zero;*
- *koniunkcja bitowa daje w wyniku liczbę, która ma bity równe jeden tylko na tych pozycjach, na których oba argumenty miały bity równe jeden (mnemonik: 1 gdy wszystkie 1);*
- *alternatywa bitowa daje w wyniku liczbę, która ma bity równe jeden na wszystkich tych pozycjach, na których jeden z argumentów miał bit równy jeden (mnemonik: 1 jeśli jest 1);*
- *alternatywa rozłączna daje w wyniku liczbę, która ma bity równe jeden tylko na tych pozycjach, na których tylko jeden z argumentów miał bit równy jeden (mnemonik: 1 gdy różne).*
- *Dodatkowo, język C wyposażony jest w operatory przesunięcia bitowego w lewo ("<<") i prawo (">>"). Przesuwają one w danym kierunku bity lewego argumentu o liczbę pozycji podaną jako prawy argument. Np. x << 2 oznacza przesunięcie o 2 pozycje w lewo.*

```c
#include <stdio.h>

void dec_to_bin(int liczba, char t[20])
{
	int i = 0, tab[16] = { 0 };

	printf("(%d)\t", liczba);
	while (liczba)
	{
		tab[i++] = liczba % 2;
		liczba /= 2;
	}
	for (int j = 15; j >= 0; j--) {
		printf("%d", tab[j]);
		if (j % 4 == 0) printf(" ");
	}
	printf("\t%s\n",t);
}
```

```
int main(int argc, char* argv[])
{
	unsigned short int a = 10, b=12, c;
	printf("%d\n", sizeof(unsigned short int));
	dec_to_bin(a,"a"); dec_to_bin(b,"b");
	c = ~a; dec_to_bin(c,"~a");			//negacja bitowa
	c = a << 2; dec_to_bin(c,"a << 2");		//przesunięcie bitowe w lewo
	c = a >> 2; dec_to_bin(c, "a >> 2");		//przesunięcie bitowe w prawo
	c = a & b; dec_to_bin(c,"a & b");		//bitowa koniunkcja
	c = a | b; dec_to_bin(c, "a | b");		//bitowa alternatywa
	c = a ^ b; dec_to_bin(c, "a ^ b");		//bitowa różnica symetryczna
	return 0;
}
```

6. Biblioteka matematyczna (funkcje i stałe matematyczne).

Aby korzystać z biblioteki standardowych funkcji matematycznych należy dołączyć ją do programu używając dyrektywy preprocesora:

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

Najczęściej wykorzystywane funkcje z biblioteki matematycznej:

`double exp(double x);`	e^x
`double log(double x);`	ln x
`double log10(double x);`	$\log_{10} x$
`double pow(double x, double y);`	x^y
`double sqrt(double x);`	$\sqrt{x}$
`double fabs(double x);`	$\lvert x \rvert$
`int abs(int i);`	$\lvert i \rvert$
`double sin(double x);`	sin x
`double cos(double x);`	cos x
`double tan(double x);`	tg x
`double asin(double x);`	arcsin x
`double acos(double x);`	arccos x
`double atan(double x);`	arctg x

Korzystając z funkcji należy przestrzegać zgodności typów argumentów!.

Stałe matematyczne (np. PI) nie są standardem języka C. Aby z nich korzystać należy dodać w poleceniach preprocesora:

```
#define _USE_MATH_DEFINES
#include <math.h>
```

7. Skompilować oraz przeanalizować przykłady do wykładu nr2

http://torus.uck.pk.edu.pl/~fialko.sergiy/text/CC/przykl/WW3.pdf

http://torus.uck.pk.edu.pl/~fialko.sergiy/text/CC/przykl/WW4.pdf

8. Napisz program, który dla danych rzeczywistych x, y, z i całkowitych k, m policzy wartości następujących wyrażeń arytmetycznych:

$$w1 = \sqrt[3]{\frac{x}{yz}} \ln(x^2 + y^2)$$

$$w2 = \sin\left(k\frac{x}{2}\right)\cos(my) + ye^{2x-1}$$

$$w3 = \left|\frac{x}{2y^2+1}\right| + \sqrt{\frac{y}{z^2+3}} + 5(y+z)^3$$

$$w4 = \frac{x}{yz}\sqrt[3]{z+1} + \sqrt[k]{x^2+z^2+1} - |y|$$

$$w5 = \frac{1}{\sqrt{x^2+y^2+k^2}} + \frac{1}{x}\sin(ky)$$

Po poprawnym skompilowaniu tego programu, przetestuj jego poprawność merytoryczną. Wyniki wydrukować w formacie long.
Wykonaj obliczenia dla: x=3.14 y=12.56 z=7 k=2 m=4.
Poprawiaj wyrażenia dopóty, dopóki nie uzyskasz następujących wyników

```
w1 = 1.68664873
w2 = 2466.40722655
w3 = 37418.15546641
w4 = -4.75167283
w5 = 0.07227756
```

9. W przykładzie WW4.pdf rozszerzyć program o wyznaczanie pierwiastków równia dla delta <0 (liczby zespolone). Przyjmijmy oznaczenia:

r1, u1 część rzeczywista i urojona pierwszego pierwiastka
r2, u2 część rzeczywista i urojona drugiego pierwiastka.

Rozwiązania równania kwadratowego można zapisać:

$$r_1 = -\frac{b}{2a}\ , \quad u_1 = +\sqrt{\left|\left(\frac{b}{2a}\right)^2 - \frac{c}{a}\right|}$$

$$r_2 = -\frac{b}{2a}\ , \quad u_2 = -\sqrt{\left|\left(\frac{b}{2a}\right)^2 - \frac{c}{a}\right|}$$

10. Dane są a,b,c - długości boków trójkąta. Sprawdzić warunek istnienia trójkąta:
https://pl.wikipedia.org/wiki/Nierówność_trójkąta
Obliczyć pole trójkąta i promień koła wpisanego w trójkąt. Wykorzystać wzory Herona:
p=(a+b+c)/2
pole: S=sqrt(p(p-a)(p-b)(p-c))
promień: R=S/p

11. Dane są cztery liczby całkowite a,b,c,d. Znaleźć wśród nich liczbę największą i wydrukować jej wartość i pozycję przyjmując, że dla a(pozycja=1), b(pozycja=2) itd. (nie wprowadzać zmiennej indeksowanej).

12. Napisać program, który będzie obliczał pole powierzchni oraz obwód takich figur jak: trójkąt, kwadrat, prostokąt, koło. Dla trójkąta sprawdzić warunek trójkąta, dla wszystkich figur wprowadzić zabezpieczenia uniemożliwiające podanie przez użytkownika wartości niepoprawnych (np. ujemnych).
Obliczenie pola kwadratu, obwodu kwadratu itd. dla wszystkich figur zapisać w osobnych funkcjach, które kolejno będą wywoływane w main().

*Treści oznaczone kursywą pochodzą z różnych źródeł internetowych.