Лекція 2. Система типів мови C++

Структура програми мовою C++, include, main(). Стандартні заголовкові файли, порядок трансляції. Система типів.
Логічний тип: зображення, оператори, вирази.
Літерний тип. Зображення літер.
Цілі типи: назви, граничні значення, зображення, оператори. Оператори присвоєння.
Дійсні типи. Перетворення типів. Математичні функції (cmath). Задача про трикутник.
Використання cin, cout.

Структура програми

Приклад: програма-привітання

first.cpp

```
#include <iostream>
#include <Windows.h>

int main()
{
	// перша "обов'язкова" програма
	std::cout << "Hello World!\n";

	// додаткові налаштування для правильного введення-виведення кирилиці
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);
	std::cout << "Введіть Ваше ім\'я: ";
	// оголошення структури для зберігання імені
	char name[20] = { '\0' };
	std::cin >> name;
	std::cout << "Hello, " << name << "!\n";
	// затримка завершення програми
	system("pause");
	return 0;
}
```

Директива *include* вказує претрансляторові долучити до тексту програми додаткові стандартні файли: *iostream* потрібен для взаємодії з потоками введення-виведення *cin* та *cout*; файл *Windows.h* містить оголошення функцій взаємодії з операційною системою, зокрема, налаштування кодової сторінки.

Функція *main* – точка входу в програму.

Дві похилі риски // позначають коментар – пояснення до тексту програми, яке не впливає на її виконання.

Стандартні імена віднесено до простору імен *std*, тому в програмі використовуємо звертання до повних (або кваліфікованих) імен, як от *std::cin, std::cout*.

Змінну *name* використовуємо для зберігання відповіді користувача (його імені), що є ланцюжком літер. Для цього використано тип «масив двадцяти літер». У оголошенні використано також ініціалізатор, що задає масив літер з кодом 0. Про оголошення і використання масивів йтиметься у наступних лекціях.

Послідовність літер, обрамлена лапками, є зображенням рядка. В окремих рядках використано спеціальні літери: '\n' – кінець рядка, '\'' – апостроф.

Виклик функції *system* просить операційну систему затримати виконання програми, щоб користувач мав час прочитати виведене на екран.

Інструкція *return* завершує роботу програми і повертає керування операційній системі. Тут 0 – код завершення, який означає, що програма закінчилася без помилок.

Алгоритм опрацювання тексту програми

1. Претранслятор заміняє в *first.cpp* директиви *include* текстами відповідних файлів і створює тимчасовий внутрішній файл (він може називатися *~first*) – одиницю трансляції. Стандартні заголовкові файли (наприклад, <iostream>) претранслятор знаходить у папках середовища програмування, а файли користувача (наприклад, "pch.h") – у папках проекту.
2. Компілятор отримує на вхід одиницю трансляції *~first*, виконує синтаксичний розбір її тексту і перетворює програму до проміжного двійкового коду *first.obj*. Якщо в програмі є синтаксичні помилки, то компілятор виявить їх, оскільки не вдасться виконати синтаксичний розбір, повідомить про них і завершить роботу.
   Створений файл двійкового коду можна побачити в папках проекту:
   /MySolution/MyProgect/Debug/first.obj або /MySolution/MyProgect/Release/first.obj
3. Редактор зв'язків отримує на вхід проміжний код *first.obj*, пов'язує виклики у ньому стандартних функцій з попередньо відкомпільованим кодом бібліотек і виготовляє виконуваний файл *first.exe*. Редактор зв'язків також може повідомити про помилку, якщо програма намагається скористатися іменем, яке не визначене ні в програмі, ні в бібліотеці.
   Готовий застосунок також можна знайти в папках проекту:
   /MySolution/Debug/first.exe або /MySolution/Release/first.exe

Типи даних

Мова програмування C++ належить до імперативних: інструкції програми вказують, як перетворити вхідні дані на результат. Програма оперує даними. Дані якого типу можна використовувати в програмі? Що таке «тип даних»?

Типом даних у програмуванні називають множину допустимих значень, для яких фіксовано спосіб кодування до двійкового вигляду і визначено перелік допустимих операцій. Спосіб кодування пов'язано з розміром пам'яті, виділеної для одного значення. Від цього розміру залежить також діапазон можливих значень типу. Тип позначають певним іменем (або конструкцією).

Вбудованим називають тип даних, інформацією про який (назва, кодування, операції) володіє компілятор. У кожній мові програмування визначено систему вбудованих типів. У мові C++ вона досить розвинена, проте й вона не покриває всіх потреб сучасного програмування. Тому C++ надає програмістові можливість оголошувати нові типи – типи користувача – відповідно до потреб задачі. Тип користувача оголошують за певними правилами на базі вбудованих типів і оголошених раніше типів користувача. Правильно сформульоване оголошення визначає множину значень і спосіб кодування нового типу. Програміст мовою C++ може визначити набір операцій для нового типу.

Система типів

- Вбудовані типи
  - Фундаментальні
    - Арифметичні
      - Інтегральні
        - Логічний – *bool*
        - Символьні – *char*, *wchar_t*
        - Цілі – [*unsigned*] *short,* ***int****, long, long long*
      - Дійсні – *float,* ***double****, long double*
    - Тип *void* – відсутні дані
  - Вказівники, наприклад *int**
  - Масиви, наприклад *char*[]
  - Посилання, наприклад *int*&
- Оголошені користувачем
  - Переліки – *enum*
  - Структури – *struct*

- Класи – *class*
- Бітові поля
- Об’єднання

Зауважимо, що логічний і літерні типи можна використовувати як різновиди цілих типів.

Докладніше див. [1, с.31-50], [2, глава 3], [5, глава 4].

Логічний тип

Логічний тип позначають словом *bool* (булевий тип, названий так на честь англійського математика Джорджа Буля). Він містить лише два значення: *false* – хиба, код 0; *true* – істина, код 1. При перетворенні числового типу до логічного всі значення, відмінні від нуля, дають *true*, нуль дає *false*. У пам’яті займає 1 байт.

Над логічними значеннями можна виконувати такі операції:

- логічні: ! (заперечення), && (кон’юнкція), || (диз’юнкція);
- порівняння: <, >, >=, <=, == (рівне), != (не рівне); *false* < *true*;
- виведення в потік <<, введення з потоку >> (вводити потрібно числа: нуль позначає хибу, не нуль – істину);
- присвоєння =, наприклад: *bool a = true*; *bool b, c*; *b = false*; *c = a && b*;
- арифметичні (бо логічний є різновидом цілого), але робити цього не варто (*true + true == true*)

Таблиця істинності логічних операторів

a	*b*	*! a*	*a && b*	*a || b*
0	0	1	0	0
0	1	1	0	1
1	0	0	0	1
1	1	0	1	1

Літерні типи

Літерні типи використовують для зберігання кодів літер. Літери потрібні для відображення рядків, текстів.

Тип *char* займає 1 байт, містить ASCII коди (діапазон від 0 до 255). Літери зображають за допомогою апострофів: 'A' – велика літера а, 'a' – мала літера а, '5' – літера (цифра) п’ять, її код відрізняється від числа 5, '+' – літера плюс, ' ' – пропуск тощо. Зауважимо, що латинська літера A та кирилична А – це різні літери, у них різні коди.

Є декілька спеціальних літер:
'\n' – кінець рядка; '\r' – перевід каретки; '\f' – кінець сторінки;
'\t' – горизонтальна табуляція (відступ на декілька літер);
'\v' – вертикальна табуляція (відступ на декілька рядків);
'\a' – звук; '\b' – повернення на одну літеру назад;
'\\' – зворотна похила; '\'' – апостроф; '\"' – лапки; '\?' – знак питання;
'\0' – літера з кодом нуль.

Тип *wchar_t* займає 2 байти, містить коди UNICODE (діапазон від 0 до 65535), використовується у застосунках для Windows. Зображення літер L'A', L'5', 'є' тощо.

Літери можна порівнювати операторами <, >, >=, <=, ==, != (порівнюють коди літер), виводити в потік оператором <<, вводити з потоку оператором >>, присвоювати =.

Цілі типи

Цілі типи використовують для зберігання цілих чисел, записаних у двійковій системі. Нагадаємо, що переведення цілого числа з десяткової системи до двійкової є точним. Типи відрізняються розміром відведеної для них пам’яті, тому можуть містити різні діапазони значень. Стандарт мови не конкретизує ці розміри, лише задає порядок «від меншого типу до більшого», вони залежать від реалізації. У таблиці наведено розміри від Microsoft. Синоніми назв типу відокремлено крапкою з комою.

Діапазони цілих типів

Назва	Розмір	Діапазон
char; unsigned char	1 байт	0..255
signed char	1 байт	–128..127
short; short int; signed short	2 байти	–32768..32767
unsigned short; unsigned short int	2 байти	0..65535
int; signed int	4 байти	–2147483648..2147483647
unsigned int	4 байти	0..4294967295 == (2^{32}–1)
long або long int == int. unsigned long == unsigned int		
long long; long long int	8 байтів	$-2^{63}..(2^{63}-1)$; $2^{63} \approx 9\times10^{18}$
unsigned long long	8 байтів	0.. 2^{64}

Зображення цілих чисел:

- звичні зі школи: 5; 2019; –29; +7 – всі типу int;
- з суфіксами, що вказують тип: 5U; 17u – unsigned; 23L – long; 5987458ULL – unsigned long long int;
- з префіксами, що вказують основу числення: 0b00001010 – двійковий, 012 – вісімковий (префікс 0), 0x0A – шістнадцятковий записи числа 10.

Оператори над цілими:

- арифметичні
  - бінарні +, –, *(множення), /(ділення), %(обчислення остачі від ділення);
  - унарні +, –; префіксні та постфіксні ++, –– (збільшення, зменшення на 1);
- порівняння <, >, >=, <=, ==, !=;
- побітові: ~ (інверсія), & (побітове і), | (побітове або), ^ (додавання за модулем 2), << (зсув ліворуч, аналог множення на 2), >> (зсув праворуч, аналог ділення на 2);
- присвоєння: звичайне (справа наліво) = ; комбіновані (спочатку дія, потім присвоєння) +=, –=, *=, /=, %=, &=, |=, ^=, <<=, >>= ;
- виведення в потік <<, введення з потоку >>.

Пояснення до операторів над цілими

2 / 3 == 0 – несподівано, але для цілих операндів ділення виконується націло.
2 % 3 == 2; 11 % 3 == 2 – звична остача від ділення цілих.
Унарний «–» змінює знак числа: *int x* = 5; *int y* = –*x*; // y == –5
Арифметичні операції бувають трьох-, двох- і одномісні:

- трьохмісне додавання (два операнди і результат) $c = a + b$; – бінарний +;
- двохмісне додавання (два операнди, результат потрапляє в лівий операнд) $S = S + x$; мовою C++ таке додавання записують комбінованим присвоєнням S += x; – «прочитати значення S, додати до нього x, результат записати в S»;
- одномісне додавання одиниці, інкремент (збільшення операнда на 1) $k = k + 1$; мовою C++ інкремент записують спеціальним оператором ++k; – префіксний інкремент;
- одномісне додавання одиниці «з пам'яттю», постфіксний інкремент k++; діє не так, як попередній, оскільки зберігає попереднє значення операнда; k++ еквівалентне виконанню групи ($t = k$, k += 1, t):
  - *int i* = 5; *int j* = ++*i*; // i == 6, j == 6
  - *int i* = 5; *int j* = *i*++; // i == 6, j == 5

Побітові оператори дають змогу маніпулювати окремими бітами числа:

- 0x13BA & 0x00FF == 0x00BA
- *int x* = 0b00110101; *int y* = ~*x*; // y = 0b11001010
- 0b00001010 << 3 == 0b01010000 // 10 * 2^3 == 80

У одній інструкції може бути декілька операторів присвоєння: $a = b = c = 5$; – всі змінні отримають значення 5.

Дійсні типи

Дійсні типи використовують для зберігання дробових чисел. Переведення їх до двійкової системи в більшості випадків є наближеним. Унаслідок цього може виявитися, що в пам'яті комп'ютера $-0{,}3+0{,}3 \neq 0$. Спосіб кодування дійсних чисел також відрізняється від цілих: зберігається мантиса і порядок числа. Мантиса $\in [0; 1)$, порядок – показник степеня числа 10, на який треба помножити мантису, щоб отримати початкове число. Такий формат називають «з плаваючою крапкою». Наприклад, $96{,}5 = 0{,}965\times10^{2}$ (тут 0,965 – мантиса, 2 – порядок); $0{,}00017 = 0{,}17\times10^{-3}$ (0,17 – мантиса, –3 – порядок). Код дійсного числа є наближеним, тому розмір виділеної пам'яті впливає на кількість точних десяткових цифр мантиси.

Діапазони дійсних типів

Назва	Розмір	Діапазон	Кількість цифр мантиси
float	4 байти	$\pm3{,}4\times10^{-38}..\pm3{,}4\times10^{38}$	до 8
double	8 байтів	$\pm1{,}7\times10^{-308}..\pm1{,}7\times10^{308}$	15
long double	8 байтів	те саме, що double	

Зображення дійсних чисел: $\pi \approx 3.14159265$ (крапка – не кома!); 10^{-6} = 1e–6 або 1E–6; 0.5 можна записати .5; дійсна одиниця 1.0 або 1.; від'ємне дійсне –1.25. Для вказання типу використовують суфікс F або L.

Оператори над дійсними:

- арифметичні
  - бінарні +, –, *(множення), /(ділення);
  - унарні +, –; префіксні та постфіксні ++, –– (збільшення, зменшення на 1);
- порівняння <, >, >=, <=, ==, !=;
- присвоєння: звичайне (справа наліво) = ; комбіновані (спочатку дія, потім присвоєння) +=, –=, *=, /=, %=, &=, |=, ^=, <<=, >>= ;
- виведення в потік <<, введення з потоку >>.

Пріоритети операторів

Пріоритет	Оператор	Асоціативність	Значення
1	`::`		Визначення діапазону доступу
	`(вираз)`		Групування
2	`()`	Л-П	Виклик функції
	`[]`		Індексування елементів масиву
	`. ->`		Прямий, опосередкований доступ до члена типу
	`++ --`		Постфіксні унарні інкремент/декремент
3	`! ~`	П-Л	Заперечення, інверсія
	`+ -`		Унарні (перед змінною, перед дужкою)
	`++ --`		Префіксні унарні інкремент/декремент
	`& * new new[] delete delete[]`		Дії з вказівниками, динамічними змінними
4	`.* ->`	Л-П	Розіменування члена класу
5	`* / %`		Арифметичні
6	`+ -`		
7	`>> <<`		Зсуви (введення/виведення)
8	`< > <= >=`		Порівняння
9	`== !=`		
10	`&`		Побітові
11	`^`		
12	`|`		
13	`&&`		Логічні

14	||		
15	=	П-Л	Присвоєння
	+= -= *= /= %= &= ^= |= <<= >>=		
16	,		Об'єднання виразів у групу

Для підвищення пріоритету виразу його беруть у круглі дужки.

Перетворення типів

[1, с.50], [2, с.115-120], [3, с.72-75]

Математичні функції

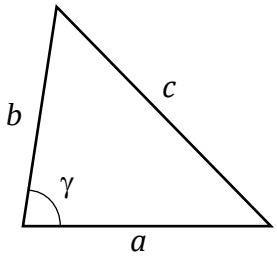
[3, с.71]

Використання cin, cout

[2, с.53, 61, 62, 63]

Задача про трикутник

Задано два дійсних числа – сторони трикутника, і ціле число – кут у градусах між ними. Обчисліть периметр, площу трикутника, радіуси його вписаного та описаного кіл.



Пригадаємо, що $S=\frac{1}{2}ab\sin\gamma$, $P=a+b+c$, $c^2=a^2+b^2-2ab\cos\gamma$, $R=\frac{abc}{4S}=\frac{c}{2\sin\gamma}$, $r=\frac{2S}{P}$. Стандартні функції обчислення синуса і косинуса приймають аргумент, заданий в радіанах, тому величину кута доведеться перетворити за формулою $\gamma_{рад}=\pi\times\frac{\gamma_{град}}{180^\circ}$.

second.cpp

```
#define _USE_MATH_DEFINES  // для використання визначеної константи PI
#include <iostream>
#include <cmath>
using namespace std;

int main()
{
	// введення заданих величин
	cout << "Input a, b: ";
	double a, b;
	cin >> a >> b;
	cout << "Input gamma: ";
	int gamma;
	cin >> gamma;
	double g = M_PI * gamma / 180.0; // перетворення кута до радіан
	double sin_g = sin(g);           // синус потрібен двічі
	double c = sqrt(a*a + b*b - 2.*a*b*cos(g)); // третя сторона за теоремою синусів
	double S = 0.5 * a * b * sin_g;             // площа
	double P = a + b + c;                       // периметр
	double R = c / (2.*sin_g);                  // радіус описаного кола за теоремою синусів
	double r = 2.*S / P;                        // радіус вписаного кола
	cout << "S = " << S << '\n'   // виведення обчислених величин
		<< "P = " << P << '\n'
		<< "R = " << R << '\n'
		<< "r = " << r << endl;
	system("pause");
	return 0;
}
```

Література

1. Бублик В.В. Об'єктно-орієнтоване програмування мовою С++.
2. Стивен Прата Язык программирования С++.
3. Дудзяний І.М. Програмування мовою С++.
4. Брюс Эккель, Чак Эллисон Философия С++.
5. Бьерн Страуструп Язык программирования С++.
6. Герберт Шилдт С++ Базовый курс.