ВСТУП. Алгоритмічна мова програмування С/С++, призначення та особливості. Історія виникнення та розвитку мови С (С++, С#).

Алгоритмічна мова програмування С/С++, призначення та особливості.

Мова програмування – штучна мова для запису команд, які виконує машина, зазвичай, електронно-обчислювальна (ЕОМ), тобто комп'ютер. Оскільки мова комп'ютера сильно відрізняється від людської мови, то розробка мови програмування (особливо на ранніх етапах розвитку комп'ютерної техніки) завжди стояла перед дилемою: створити мову зрозумілу для людини, чи зручну для ЕОМ. Тому, ще за часів створення перших ЕОМ мови програмування почали поділяти на ***машинні***, ***машинно-орієнтовані***, та ***алгоритмічні*** (мови програмування високого рівня).

Алгоритмічні мови програмування використовують синтаксичні конструкції, що містять слова та вирази людської мови і є більш зручними для написання коду людиною. При цьому для виконання програми, записаної алгоритмічною мовою, її потрібно транслювати в послідовність зрозумілих для комп'ютера команд. Для цього використовують спеціальні програми – транслятори. Транслятори розробляються і вдосконалюються разом з мовою програмування і бувають двох типів: ***інтерпретатори*** і ***компілятори***.

Інтерпретатор транслює команди програми, читаючи і перекладаючи їх послідовно одна за одною. Команда перекладається на мову комп'ютера та виконується, у випадку її успішного виконання інтерпретатор переходить до опрацювання наступної команди. Процес трансляції коду програми інтерпретатором називаємо ***інтерпретацією***, наприклад, команди сценаріїв JavaScript інтерпретуються і виконуються і процесі завантаження вебсторінки, на якій вони записані. Інтерпретатори простіші у виготовленні та використанні, проте виконати програму на інтерпретованій мові програмування у випадку відсутності чи вимкнення інтерпретатора не вдасться.

Процес трансляції, при якому код усієї програми відразу транслюється в окремий виконавчий модуль називаємо ***компіляцією***. Беззаперечною перевагою ***компіляторів*** є те, що готовий скомпільований код можна багаторазово виконувати навіть за відсутності компілятора. ***Компіляцію*** застосовують для розробки програмного забезпечення, що може працювати окремо від середовища його розробки, більшість програм нашого ПК написані, скомпільовані і надані нам для використання вже в готовому вигляді. Щоправда, розробка компіляторів є складнішою через потребу в додаткових аналізаторах програмного коду на наявність помилок.

Мова С (читаємо «Сі») є алгоритмічною мовою програмування, що підтримує парадигми ***імперативного*** (програма з інструкцій, що є командами для виконання) та ***структурного*** (програму можна розділяти на окремі підпрограми) програмування. Хоча сама мова С є машинно-незалежною

мовою програмування високого рівня, для її створення було використано групи команд машинно-орієнтованої мови Assembler. Це дозволило поєднати переваги машинно-орієнтованих мов в плані максимального використання можливостей конкретної обчислювальної архітектури з перевагами алгоритмічних мов програмування. Ця особливість мови С зробила її у свій час найпопулярнішим засобом розробки програмного забезпечення, як прикладного, так і системного.

Алгоритмічна мова С++ виникла на основі мови С шляхом додавання до неї елементів ***об'єктно-орієнтованого програмування (ООП)***. Про це свідчить навіть її початкова назва "Сі з класами" (клас – одне з фундаментальних понять ООП), яку дав їй її засновник Б'ярне Страуструп. Увібравши разом з базовим функціоналом усі переваги своєї попередниці С++ перейняла і її провідну роль в розробці програмного забезпечення на наступному етапі розвитку технологій програмування. Цьому етапу притаманна потреба в ООП для оптимального структурування значного обсягу програмного коду, що власне і спонукало перехід розробників від С до С++. Об'єктно-орієнтоване програмування і далі використовується більшістю сучасних технологій в якості ключової парадигми програмування.

Сьогодні мови програмування С та С++ є двома цілком самостійними мовами програмування, які існують і розвиваються незалежно одна від одної. Вживання в цьому навчальну посібнику дещо штучного поєднання "мова програмування С/С++" має конкретну навчальну мету та є оправданим з огляду на історію створення С++ так і на спільність базових синтаксичних конструкцій. Ця спільність знаходить себе навіть в тому, що деякі автори навчально та довідкової літератури з С++ схильні ділити матеріал на дві частини: "підмножина С", куди входить базовий синтаксис, та функціонал структурного програмування та "підмножина С++", що містить опис реалізованого в С++ ООП.

Інтегроване середовище розробки ***Microsoft Visual Studio***, що використовується для практичної частини цього курсу, також поєднує обидві мови програмування. Visual Studio має лише компілятор для компіляції програм, написаних на С++, для компіляції ж коду на "чистому" С (ANSI C) середовище використовує цей же компілятор із спеціальними налаштуваннями.

Загальновідомою є роль мов програмування С та С++ у створенні інших сучасних технологій розробки програмного забезпечення. Так наприкінці минулого століття, правонаступниця синтаксису мови програмування С, найпопулярніша на той час технологія розробки системного і прикладного ПЗ С++ передала цей програмний синтаксис таким новітнім, уже мережевим, технологіям розробки, як Java, технологія серверних скриптів PHP, мова сценаріїв JavaScript, мова програмування C# в технології .net від Microsoft. З огляду на це, вивчення студентами основ програмування саме на "мові С/С++" покликане надати студентові базу для успішного вивчення будь-якої з перелічених вище технологій у разі виникнення в нього такої потреби.

Структура простої програми на мові С.

Для того, щоб з'ясувати базові засади написання програмного коду на мові С, розглянемо наступну програму:

```
1   #include<stdio.h>
2   int main()
3   {
4           int a, b, sum;
5           printf("Enter two int number:\n");
6           scanf_s("%d%5d", &a, &b);
7           //scanf("%5d%5d", &a, &b);
8           sum = a + b;
9           printf("%5d + %5d = %5d\n", a,b,sum);
10          return 0;
11  }
```

У першій стрічці коду підключається так званий заголовковий файл (header-файл), що містить потрібні для роботи коду бібліотечні підпрограми (в нашому прикладі – функції ***printf*** та ***scanf_s***). При написанні програми розробник не повинен вдаватися до таких подробиць як, наприклад, ввід чи вивід букв і цифр на фізичному рівні обладнання. Для цього кожна мова програмування надає ***бібліотеку стандартних підпрограм***, в якій уже реалізовано обробку низькорівневих процесів, а програмісту для їх організації достатньо звернутися до потрібної підпрограми. Бібліотечні функції С зберігаються в окремих файлах з розширенням ***.h*** (скорочення від header) і підключаються до програми директивою ***#include***.

Друга стрічка містить оголошення (сигнатуру) функції ***main***, з якої буде розпочинатися виконання програми. Структурна парадигма мови С струнка:

- програма складається з функцій, кожна функція має унікальне ім'я;
- першою виконується функція з назвою ***main;***
- інші функції можуть виконуватися лише, якщо їх викликати з коду програми;
- файл програми може містити декілька функції, але жодна функція не може міститися в тілі іншої функції.

Перелічені правила описують далеко не усі засади написання і виконання програми на С. Знайомитися з іншими будемо в ході нашого курсу далі.

Заголовок функції у стрічці 2 містить, власне, ім'я функції ***main,*** перед яким вказується тип результату цієї функції. В нашому прикладі функція у результаті повинна давати ціле число, про що вказує тип ***int*** перед її назвою. Круглі дужки після назви функції призначені для переліку параметрів, що

передаються у функцію. В нашому випадку, функція ***main*** не отримує жодних параметрів при запуску, тому дужки порожні.

Фігурна дужка в третьому рядку вказує на початок, тіла функції, яке завершується закриваючою фігурною дужкою в рядку 11. В тілі функції записують команди програми, що виконуються при виклику функції.

Перша команда функції оголошує три змінні цілого типу. Оголошення змінних зобов'язує компілятор виділити певну кількість байтів пам'яті для зберігання даних і позначити її вказаними іменами змінних. Кількість пам'яті, потрібна для зберігання даних, визначається типом, назва якого записана перед іменами змінних. Оголошувати змінні можна у будь-якому місці програми, але до їх першого використання.

У стрічці 5 викликається функція ***printf*** для виводу повідомлення на консоль (екран ПК). Текст повідомлення записано в лапках в дужках після назви функції. Текст завершується символами "**\n**", що виводять на консоль спеціальний символ закінчення абзацу, тобто далі текст на консолі почне виводитися з нового рядка.

Функція ***scanf_s*** у 6-му рядку виконує читання двох цілих чисел з консолі (з клавіатури, але набір чисел на клавіатурі супроводитиметься виводом на екран) та розміщує їх у змінні a та b. Для запису даних у змінну програмі потрібно знати місце в пам'яті ПК, де змінна зберігає свої дані (***значення***). Для цього функція ***scanf_s*** отримує не самі назви змінних, а їх адреси у пам'яті комп'ютера, на що вказує знак "&" перед іменем змінної.

Сьомий рядок містить коментар. Рядок програми, або його частина, що починається подвоєним знаком "слеш" вважається компілятором за коментар і компілятор не транслює його у результуючий код програми. Це використовують для розміщення в коді програми пояснень до нього, а також для відключення окремих рядків самого коду від виконання.

Закоментований в рядку 7 виклик функції ***scanf*** має на меті звернути увагу читача на те, що саме ця функція використовується за правилами мови С. Вживання замість ***scanf*** функції ***scanf_s*** пов'язане з тим, що використання функцій з прямий доступ до пам'яті ПК може бути не зовсім безпечним для самого комп'ютера. Розробники Microsoft в нових версіях Visual Studio додають до бібліотек "безпечні" (safe) аналоги функції, що вимагають доступу до пам'яті, і заохочують їх використання програмістами замість стандартних.

У стрічці 8 записано оператор присвоєння "**=**", який отримує значення змінних ***a*** та ***b***, додає ці значення і результат виразу записує у змінну ***sum***.

В дев'ятій стрічці, так само, як у стрічці 5, програма виводить повідомлення на екран за допомогою функції ***printf***. Але цього разу текст повідомлення записаний в лапках є шаблоном, в який вписуються значення змінних, що слідують після коми. Місця, куди вставляються ці значення позначені символами "**%5d**". Вони заповняються у тій же послідовності, в якій записані змінні після коми, а сам запис (форматний вказівник) "**%5d**" означає, що виводиться п'ятицифрове ціле число (decimal). Якщо числа матимуть

менше цифр, то їх доповнюють пропусками зліва, якщо більше, то виводяться всі цифри всупереч вказаному формату.

Завершується тіло функції оператором (командою) **`return`** в стрічці 10. Ця команда виконує дві дії:

- завершує виконання функції, тобто, якщо записати після цієї команди ще який програмний код, він не виконуватиметься ніколи;
- повертає результат роботи, тобто значення, яке відповідає типу, вказаному у заголовку функції, в місце її виклику.

Щодо результату функції **`main`**, то існує домовленість, за якою число 0 у результаті її виконання вказує на успішне завершення програми.

Історія виникнення та розвитку мови С (С++, С#).

Історія виникнення мови програмування С пов'язана із співробітником американської фірми Bell Labs Денізом Рітчі. На саму назву мови вплинули розробки та інших системних програмістів (М. Річардс - система BCPL, К. Томпсон - мова В). Створення різних програмних засобів, що полегшували б роботу системних програмістів, було спричинено розпочатою в 1969 році розробкою операційної системи Unix для комп'ютера PDP-7. На той час єдиною операційною системою великого комп'ютера GE-645, що обслуговував співробітників лабораторії, була досить громіздка багатокористувальницька система Multics. Кен Томпсон (один з розробників Multics) в свій час написав програму, яка моделювала рух небесних тіл. Кожен її запуск на GE-645 обходився в 75 $, а траєкторії руху видавалися в табличному вигляді.


Kenneth Lane Thompson

Тоді невелика група співробітників, очолювана К. Томпсоном, вирішила створити більш зручну однокористувальницької систему на маленькому покинутому комп'ютері PDP-7 з дисплеєм. До складу цієї групи входив і Деніз Рітчі. Система Unix стала дуже популярною серед співробітників лабораторії, тому, що вона суттєво спрощувала процес проходження завдань і не вимагала від

користувачів знання численних директив системи Multics. У 1970 році Д. Рітчі допоміг перенести Unix на більш потужний комп'ютер PDP-11. У процесі цієї роботи став у нагоді набір макрокоманд асемблера, який спрощував програмування численних процедур. Цей набір і був покладений в основу мови С, яка вдало поєднала специфіку машинних команд з елементами мови високого рівня. У 1973 році Д. Рітчі і К. Томпсон переписали ядро операційної системи Unix на мову С (до цього всі програми були написані на асемблері).

З 1974 року система Unix разом з вихідними текстами на мові С і компілятор цієї мови були передані низці американських університетів. Значну роль у подальшому розвитку системи Unix, що перетворилася з однокористувальницької на багатокористуваць ку, відіграли співробітники університету Берклі. Популярність системи Unix, що збереглася до наших днів у різних версія свого наступного покоління, під спільною назвою Linux і обслуговує сьогодні близько 90% серверів, значною мірою сприяла і популярності мови С, компілятор якої входив до складу Unix.


Bjarne Stroustrup

Наступний крок у розвиток потужності та універсальності мови С в 1983 році вніс співробітник все тієї ж Bell Labs Б'ярне Страуструп (Bjarne Stroustrup). Запропоновані ним об'єктно-орієнтовані розширення мови С поклали початок нової мови програмування – С++ (перша назва – С з класами). Ці удосконалення дали користувачам можливість конструювати власні типи даних, додавати до мови нові операції над такими даними, агрегувати дані з методами їх опрацювання, успадковувати і перевизначати методи в похідних класах.

Слід зазначити і істотний внесок в розвиток систем програмування на базі мов С/С++, внесений фірмою Borland, точніше, її засновником - Філіпом Канном. Це створення інтегрованих систем розробки, в яких вдало поєдналися засоби підготовки, зберігання, налагодження та компонування програм, які спочатку з'явилися в системі Turbo Pascal. Всі пізніші системи програмування в тій чи іншій мірі запозичили основні ідеї Ф. Канна. Подальший розвиток систем програмування на базі мови С++ пов'язаний з Borland C++, Borland C++ Builder, Visual C++ фірми Microsoft, Microsoft Visual Studio і ін.

У 90-х роках минулого століття розробка програмного забезпечення виходить на якісно новий рівень, пов'язаний не тільки інтенсивним розвитком комп'ютерної техніки, а й з потребами комп'ютерних мереж, що впроваджуються в найрізноманітніші сфери людської діяльності. В цей період закладено підвалини таких відомих технологій програмування, мова серверних скриптів для програмування динамічних веб-сторінок PHP, технологія скриптів клієнтської частини JavaScript, технологія розробки кросплатформних та мережевих додатків Hot Java з мовою програмування Java. Найпопулярнішою серед програмістів, як системних, так і прикладних, мовою програмування на той час була С++. З огляду на це розробники згаданих вище

технологій брали за основу С++, намагаючись полегшити програмістам їх освоєння. Таким чином значна частина сучасних засобів розробки ПЗ отримувала власні мови програмування на основі так званого "С-подібного" синтаксису.

Вже на початку XXI-го століття компанія Microsoft, яка на той час активно конкурувала з Borland за провідну роль у процесі розвитку С++, успішно започаткувала ще одну сучасну платформу для розробки програмного забезпечення – ***.net*** (читаємо "***дот нет***"). Нова технологія значною мірою враховувала досвід використання Java, зокрема, вдосконалила ідею проміжного середовища (***віртуальної машини***) для виконання java-програм, яке працює поверх операційної системи. Але розробники Microsoft пішли далі, – їхня віртуальна машина CLR (Common Language Runtime) забезпечує виконання програм написаних на різних мовах програмування.

"Багатомовність" ***.net*** дозволила Microsoft залучити в свою нову технологію популярні на той час мови програмування Visual Basic так С++. Крім того, спеціально для ***.net*** була розроблена абсолютно нова мова програмування ***C#*** (читаємо "***сі-шарп***"). Зі своїми попередницями С та С++ нову мову зв'язує не тільки спільна назва, але й уже згаданий "С-подібний" синтаксис та багато іншого. До прикладу, відомий механізм С++ перевизначення арифметичних, логічних та інших операторів для нових типів даних на основі класів, від якого (з огляду на його низьку практичну цінність) відмовилися розробники Java, зберігся в С#. Концептуально ж, мова С# побудована повністю на об'єктно-орієнтованій парадигмі, що робить її ближчою до Java, аніж до С чи С++.