Основни понятия

Компютър - устройство, което може да бъде програмирано да извършва последователност от аритметични и логически операции.

Програма - списък от инструкции отправени към компютъра, които трябва да дадат определен изходен резултат. Четима от човека част се нарича source code, от който програмите се компилират, дава възможност за изследване протичането на програмата.

Компилатор - програма, превеждаща даден изходен код (програма) до език от по-ниско ниво (машинен или асемблерен език). Компилираният код може да се изпълни на нашия компютър. Има две основни фази при създаване на кода: аналитична и синтезираща

Аналитична - програмата се разбива на токени (ключови думи, идентификатори и оператори). Проверява се дали граматиката на езика е спазена (използва се дървовидна структура). Проверява се дали токените са логически свързани помежду си (примерно дали променлива е дефинирана преди да бъде използвана).

Синтезираща - създава се междинен код. Всичко се оптимизира (бързината или използването на оперативна памет се подобрява). Превежда се дървовидната структура, ако всичко е минало успешно.

Интерпретатор - програма, изпълняваща друга програма. Извършва се последователен анализ на командите от програмата ни, превежда се на машинен език и се изпълнява, тоест не превежда целия код само веднъж,

както при компилатора, а при всяко извикване на програмата ни. Работи по-бавно от компилатора.

Разлики между двете - при използване на компилатор, при всяка промяна в кода трябва да изчакаме компилатора да транслира всички файлове и да ги свърже. Интерпретаторът превежда само променената част от кода, което изисква значително по-малко време. Вече компилиран код се нуждае от по-малко време за стартиране, отколкото интерпретирането му, тоест интерпретаторът трябва отново да анализира всичко, докато компилираният код само изпълнява логическите операции в последователност.

Кратко видео, демонстриращо разликата между двете: https://www.youtube.com/watch?v=_C5AHaS1mOA

Азбука - крайна редица от допустими символи, с помощта на които се постигат основните конструкции на езика. (a - z, A - Z, 0 - 9, + , - , * , / , = , (,) , [,] , { , } , ; , : , , , “ , ‘ , < , > , . , _ , ! , @ , # , $, % , ^ , ~ , |

Синтаксис - множество от правила, които определят образуването на основните конструкции на езика на база на азбуката му. Не всяка редица от символи е програма.

Семантика - правила, които определят как се изпълнява програмата в даден език за програмиране.

Метаезик на Бекус-Наур - отделните категории се отбелязват заградени в <> (<число>, <оценка>, …). Знакът ‘|’, или pipe означава ‘или’ (<цифра> ::= 0 | 1 | .. | 9). Знакът {...} означава че съдържанието в скобите се повтаря 0 или

повече пъти (<списък_от_оценки> ::= <оценка>{, <оценка>}. Знакът [...] показва, че съдържанието не е задължително (<цяло_число> ::= [+|-]<число_без_знак>)

Сведения за C++

Case Sensitive - езикът прави разлика между малки и главни букви, тоест съществува различие. (name != Name != NAME), където комбинацията от символите “ != ” означава е различно.

Идентификатори - за именуване на променливи, константи, типове данни и други съществуват следните правила: трябва да започват с буква и да съдържат цифри, букви и долни черти. (Правилно именуване на променлива: min, max, index_, MIN_SIZE). (Грешно именуване на променлива: 1size, razmer, malko.dulgo-ime, сума).

Запазени думи - идентификатори, чиито значение е фиксирано от езика и не може да бъде променяно (if, else, switch, while, for, break, continue, int, double, ...).

Стандартни думи включени в стандартните библиотеки на езика, които имат специално значение, но могат да бъдат предефинирани (cout, cin, ceil, sqrt, floor, ...).

Променлива - място за съхранение на данни, което може да съдържа различни стойности по време изпълнение на програмата. Характеризират се по следния начин, като имат: (място в паметта, идентификатор, тип, стойност). Преди да бъдат използвани, трябва да бъдат дефинирани с конкретен тип, понеже C++ е строго типизиран език.

Константи - променливи, чиято стойност бива зададена веднъж, след което не може да бъде променяна. (const int size = 64)

Коментари - части от кода, които биват игнориране от компилатора и интерпретацията, като случат за конкретизиране и обяснение на дадена част от кода, тоест за лесна четимост на програмата. В C++ има два начина за закоментиране на текст (// коментар, /* коментар */).

Деклариране на променлива - <променлива> ::= <тип> <име> (int age, double average, …).

Команди за вход и изход от конзолата - (cin >> <променлива>, cout << <променлива>), задължително трябва променливата да бъде дефинирана преди използването и). Командите са дефинирани в библиотеката <iostream>. За добавянето на нов ред се използва endl.

Числени типове и операции с тях

- int - целочислен тип, с допустими стойности в интервала [-2 ^ 31; 2 ^ 31 - 1], като заема 4 байта памет.
- double - представя реално число в интервала [-1,74 * 10 ^ 308; 1,74 * 10 ^ 308], като заема 8 байта памет.
- float - реално число в интервала [-3,4 * 10 ^ 38, 3,4 * 10 ^ 38], което заема 4 байта памет.

Допустимите операции с гореспоменатите типове са: +, -, /, *, %. Оператори за сравнение дефинирани за числата са: >, <, >=, <=, ==, !=. Унарни операции са: +, -. За използването

на вече дефинирани математически функции се използва библиотеката <cmath>.

Символен тип - използва се за представяне на знаци като букви, цифри, препинателни знаци. Представлява множество от стойности в интервала [-128, 127], като заема 1 байт памет. Всеки символ има своя уникална стойност спрямо ASCII таблицата (стойността на ‘A’ е 65). Тоест всеки знак бива представен някак си чрез число.

Булев тип - логически тип, пазещ в себе си логически стойности, които биват или лъжа, или истина (true | false). Всичко се представя чрез 0 - false, 1 - true. В паметта заемат 1 байт.

Преобразуване от един към друг тип - така нареченото кастване се използва за преминаване между различни типове. Примерно ако имаме символ (int number = 10) и се нуждаем да изкараме резултат от някакво деление, който не е цяло число, то ние трябва да предимен към реален тип - double). Всичко това се получава така : std::cout << (double)number / 4 << std::endl; Полученията резултат ще бъде 2.5, а ако нямаше това кастване щеше да бъде 2 защото има загуба на информация при делене на целочислени типове. Загуба на точност има и при преминаване от по-висок към по-нисък тип (double -> int, float -> int, ...).

Логически операции

- && (and) - връща истина, само когато и двата аргумента на оператора са истина.
- || (or) - връща истина, когато поне един от аргументите е истина.

- ! (not) - връща истина, ако аргумента е лъжа и обратното.

x	y	x && y	x || y	!x
true	true	true	true	false
true	false	false	true	false
false	true	false	true	true
false	false	false	false	true

Математически операции: (#include <cmath/math.h>)
abs(x) абсолютна стойност, x е целочислен израз
fabs(x) абсолютна стойност, x е реален израз
sin(x), cos(x), tan(x), asin(x), acos(x), atan(x) exp(x),
log(x) – натурален, log10(x) - десетичен логаритъм
ceil(x) – закръгляне нагоре, floor(x) – закръгляне надолу
sqrt(x) – корен квадратен, pow(x, n) – x на степен n