Упражнение 02
Първи стъпки

Какви са стъпките при създаването на една програма? Да се напомнят някои от правилата за конструиране на функцията main()... Да се припомни за входно-изходните операции...

Прости програми, които изискват единствено четене на данни от входа, прости операции върху тях и след това извеждане на екрана...

Създаване на проста програма на C++

Език за програмиране: синтаксис и семантика. Синтаксисът определя елементите, които могат да се използват за създаването на програма и правилата, по които те трябва да се комбинират, за да се получи валидна програма. Не всяка редица от символи е валидна програма... Семантиката е значението, което се влага във всеки един от елементите на програмата.

Етапи на създаване на програма

1. Текстът на програмата, кодът, се записва във файл. Това е т.нар. **изходен файл** (source file). Какво използваме за създаването на файловете с изходен код?
2. Компилиране на изходния код. Стартира се програма, наречена **компилатор**, която превежда изходния код до код на машинен език. Файлът, който съдържа преведената програма, е **обектния код** на програмата. Обектен код се генерира само за валидна програма, т.е. компилаторът проверява дали кодът отговаря на всички правила, описани в синтаксиса на езика.
3. Свързване на обектния код с допълнителен код. Процесът се осъществява от т.нар. **свързваща програма linker.** Програмите на C++ обикновено използват библиотеки. Библиотеките съдържат обектен код на подпрограми, които извършват например извеждане на информация на екрана, прочитане на данни от клавиатурата, математически функции за повдигане на степен, намиране на корен квадратен... Файлът, който съдържа обединените обектни кодове се нарича **изпълним.**

Интегрирана среда за разработка (Integrated Development Environment IDE)

Средите за разработка автоматизират всички стъпки по създаването на програмата и предоставят удобен графичен интерфейс. Т.е. освен текстов редактор, в който да бъде описан изходния код, има вградени компилатор, линкер, дебъгер...

Кои среди за разработка ще бъдат използвани по време на упражнения? Microsoft Visual Studio, Code::Blocks...

Обикновено, когато се създава изходния файл, той е с разширение .cc, .cxx, .cpp. Разширението дава информация за езика, който е използван за създаването на програмата. Така средата може да избере подходящ компилатор.

Да се създаде нов C++ проект. За сега Console Application... Да се разгледат командите в меню Build...

Compile се отнася до процеса на компилиране на текущия файл, с който се работи в момента.

Build компилира всички изходни файлове, които са част от проекта. Ако в проекта са включени няколко файла, компилират се само тези, които са били променени.

Link стартира свързващата програма, която обединява компилирания изходен код с кода на библиотеките.

Run стартира програмата. Ако предходните стъпки не са изпълнени, командата Run ще ги изпълни, преди да стартира програмата.

Debug е команда, която стартира постъпковото изпълнение на програмата. Обикновено се използва в процеса на търсене и отстраняване на логически грешки в програмата. Програмата се трасира, за да се открие коя операция предизвиква грешка.

Някои правила при конструиране на програма на C++
Всяка програма на C++ е съставена от една или няколко функции. Главната функция, която задължително трябва да присъства в програма на C++ е функцията `main()`.

Функцията е последователност от оператори, които изпълняват определени действия. Когато операционната система стартира програмата, като извиква функцията `main()`, това е входната точка на програмата. Функцията изпълнява своите оператори и връща някакъв резултат към операционната система.

```
int main()
{
    return 0;
}
```

При описването на функцията трябва да бъдат включени няколко **задължителни** елемента.

- Тип на резултата, който връща функцията. Функцията `main()` трябва да върне като резултат цяло число. Операционната система използва този резултат, за да определи дали програмата е завършила нормално или е възникнала грешка. Типът на резултата се описва с вградения тип данни `int`.
- Име на функцията. **C++ е чувствителен към малки и главни букви**, така че името на функцията трябва да бъде изписано много точно.
- Списък с входни параметри на функцията. Този списък може да бъде празен, но **задължително** трябва да има включени **отварящa** и **затварящa кръгли скоби ()**.
- Тяло на функцията. Тялото на функцията се състои от отделни инструкции (операции), които се описват в блока между **отварящата и затварящата фигурни скоби {}**. Единственият оператор в тялото на функцията е операторът **`return`**, с който завършва кода на функцията. С операторът **`return`** се указва какъв е резултатът от изпълнението на функцията. Ако функцията върне като резултат 0, това е знак за операционната система, че изпълнението е протекло нормално. Всяка друга стойност се интерпретира като възникване на определена грешка.

Операторите в C++ трябва да завършват с ;. Това е често срещана грешка по време на компилация.

Вход и изход
В самия език C++ няма дефинирани оператори за вход и изход. Те са дефинирани в библиотека. Т.е. за да може да се прочете някаква информация от клавиатурата или да се изведе някаква информация на екрана на компютъра, трябва изрично да бъде включена тази стандартна библиотека `iostream`.

Обектът, който е свързан с клавиатурата и се използва за четене на входни данни е `cin`, нарича се още стандартен вход.

Обектът, който е свързан със стандартния изход – екрана е `cout`.

Задача 1:
Да се създаде програма, която намира сумата на две цели числа.

Директивата `#include` и стандартното именно пространство
Първият ред от програмата `#include <iostream>` е директива към препроцесора, която указва на компилатора, че библиотеката `iostream` трябва да бъде включена. Използва се директивата `#include`, последвана от ъглови скоби, в които се посочва заглавния файл (*header*) на библиотеката. В изходния код на програмата могат да бъдат включени повече от една библиотеки. Като правило директивите за тяхното включване се поставят в началото на изходния файл.

Обектите `cout`, `cin` и `endl` са дефинирани в рамките на т.нар. стандартно именно пространство `std` (namespace). Обикновено именните пространства се използват, за да бъдат избегнати конфликти, причинени от съвпадения на имена в различни библиотеки.

```
#include <iostream>

using namespace std;

int main()
{
    cout << "Please, enter two integer numbers: " << endl;

    int a, b;
    cin >> a >> b;

    cout << "The sum of " << a << " and " << b << " is "
         << a + b << endl;

    return 0;
}
```

Оператор за изход

Извеждането на информация на екрана се извършва с оператора <<. Операторът е бинарен и инфиксен. Изисква два аргумента: стандартния изход (или обекта `cout`) и информацията, която трябва да се изведе. Операторът се изпълнява отляво надясно. Защо е възможно това каскадно извикване на оператора за изход? След приключване на изпълнението, операторът връща като резултат своя ляв операнд, който е обекта `cout`.

Текстът, който трябва да се изведе на екрана се поставя в кавички „“. Нарича се още *низов литерал*.

Какво е `endl`? Нарича се *манипулатор*. В резултат от неговото използване, извеждането на екрана преминава на нов ред. Какво друго може да се използва за нов ред? ‘\n’.

Оператор за вход

Преди да се извърши четенето от стандартния вход, трябва да се определи къде ще бъдат записани въведените от клавиатурата данни.

Декларират се две целочислени променливи `a` и `b`, като първо се обявява какъв е типа на данните, които ще бъдат записани в тях, а след това и техните имена.

Четенето на входни данни от клавиатурата се извършва с оператора за вход >>. Отново, бинарен и инфиксен оператор. Левият операнд е обектът `cin`, свързан със стандартния вход. Десният операнд е променливата, в която трябва да бъде записана, въведената от клавиатурата стойност.

Отново операторът е лявоасоциативен и е възможно каскадно извикване, защото резултатът от неговото изпълнения е левият операнд, обектът `cin`.

За да се инициализира всяка от променливите в обръщенията към оператора за вход >>, данните от клавиатурата трябва да бъдат разделени от интервал, табулация или нов ред.

Какво ще се промени в резултата от изпълнението на програмата, ако извеждането се осъществи по следния начин:

```
cout << "The sum of a and b is " << a + b << endl;
```

Коментари в C++

Коментарите служат за пояснения на определени действия. Колкото по-сложни са програмите, толкова по-голяма е нуждата от коментари. Полезни са не само за други програмисти, които разглеждат кода, но и на създателя на кода, ако след време се наложи ревизия. Коментарите се пренебрегват от компилатора, заменят се с интервал при процеса на компилиране.

Коментарите в C-стил са многоредови коментари, които не позволяват влагане. Започват с /* и завършват с */.

Коментарите в C++ стил започват с две наклонени черти // и приключват в края на реда.

Да се кажат няколко думи за форматирането на кода...

Деклариране на променлива, инициализация и оператор за присвояване

Деклариране на променлива се състои от две части:

- тип на данните, които ще бъдат съхранявани в променливата;
- името, с което тези данни могат да бъдат достъпени.

```
int number;
```

Тип `int` е вграден тип данни, който съответства на целите числа, числа без дробна част. Према както положителни, така и отрицателни стойности. Диапазона на допустимите стойности зависи от реализацията на езика... В следващото упражнение по-подробно ще бъдат разгледани типовете данните.

Втората част от декларацията е името на променлива. В случая променливата е `number`. Защо променлива? Защото нейната стойност може да се промени в хода на програмата. Всяка променлива трябва да бъде декларирана преди да се използва.

Декларацията завършва с ;.

На променливата може да се присвои (даде) стойност в момента на нейното деклариране. Тогава променливата е **инициализирана** с начална стойност.

Оператор за присвояване

За да се даде стойност на променливата при първоначалната инициализация или да се промени стойността на променливата в хода на програмата се използва операторът =. Операторът = се нарича оператор за присвояване.

```
int number; // неинициализирана променлива

int number = 123; // инициализирана променлива

// използва се операторът за присвояване,
// за да се промени стойността на променливата
number = 23;
```

Операторът за присвояване е бинарен, инфиксен. Изпълнява се от дясно наляво.

```
a = b = c = 0;
```

Оценката на израза `c = 0` е стойността `0`, след това тази стойност се присвоява на `b` и накрая и на променливата `a`.

Каква е оцената на следния израз?

```
number = number - 1;

// може да се замени с оператора --, едноаргументен
number --;
```

Кратък преглед на операторите в C++

Категория	Оператори
Аритметични	`+, -, *, /, %` `++, --`
Логически	`&&, ||, !`
За сравнение	`>, >=, <, <=, !=, ==`

Внимание! Много често срещана грешка е използването на оператор за присвояване = вместо оператор за сравнение ==.

Закони на ДеМорган

```
!(a && b) = !a || !b
!(a || b) = !a && !b
```

Когато се оценява израз с дизюнкция, ако първият операнд има стойност истина, вторият изобщо няма да бъде оценен. Стойността на целия израз ще бъде истина.

Когато се оценява израз с конюнкция, ако първият операнд има стойност лъжа, вторият изобщо няма да бъде оценен. Стойността на целия израз ще бъде лъжа.

Побитови операции

(може да се разкаже към края на упражнението, след като минат задачите)

В компютрите всички данни, в частност числата, се представят като поредица от 0 и 1. Т.е. числата се представят в двоична бройна система. Побитовите операции действат върху двоичното представяне на числата.

Също като логическите оператори, в C++ има побитово „и“ (&), побитово „или“ (|), побитово отрицание (~) и побитово “изключващо или” (^).

x	y	~x	x & y	x | y	x ^ y
0	0	1	0	0	0
0	1	1	0	1	1
1	0	0	0	1	1
1	1	0	1	1	0

Съществуват още два логически оператора, които нямат съответствие при логическите оператори. Това са побитово отместване наляво (<<) и побитово отместване надясно (>>).

Операторите са бинарни. Левият операнд е променливата, която искаме да променим. В дясно от оператора е броя на позициите, с които искаме да отместим битовете.

Например:

```
int number = 3; // 00000011

int numberShiftLeft = number << 2; // 00001100
int numberShiftRight = number >> 2; // 00000000
```

Проверка за четно число

При нечетните числа младшият бит е 1. При четните числа младшият бит е 0. За да се провери стойността на младшия бит, използваме операцията побитово „и“ с 1. Това ще премахне всички останали битове, освен младшия (единицата го запазва).

```
int number = 3; // 00000011

00000011 &
00000001
00000001
```

Ако резултатът, който се получава е 0, числото е четно. Ако резултатът е 1, то числото е нечетно.

Как да се провери какъв е бита на позиция n?
За да се намери бита на позиция `n`, трябва да се нулират всички битове, освен бита на позиция `n`. За целта може да се използва маска, която на желаната позиция има стойност 1, а във всички останали позиции има стойност 0. Операцията, която трябва да се приложи е побитово „и“. Как се конструира маската?

```
int position = 3;
int mask = (1 << position); // 00001000

int bitAtPosition = number & mask;
bitAtPosition = bitAtPosition >> position;
```

Стойността на променливата `bitAtPosition` преди прилагането на операцията `>>` е всъщност `2^position`.

Нулиране на бита на позиция n?
За да се нулира бита на позиция `n`, към него трябва да се приложи операция побитово „и“ с 0. За да се запазят всички останали битове, към трябва да се приложи същата операция, но с 1.

Необходимата маска може да се получи по следния начин:

```
int position = 3;
int mask = ~(1 << position);

// ~(00001000) = 11110111
int result = number & mask;
```

Установяване на n-тия бит в стойност единица
Всички позиции, освен желаната, трябва да се запазят. Позиция `n` трябва да приеме стойност 1.

Операцията трябва да бъде побитово „или“, а маската да има 0 във всички позиции, освен позиция `n`.

```
int position = 3;
int mask = (1 << position);

int result = number | mask;
```

Инвертиране на n-тия бит

```
int position = 3;
int mask = (1 << position);

int result = number ^ mask; // 0 ^ 0 = 0, 1 ^ 0 = 1
```

Ако `n`-тия бит е бил 1, когато се приложи операцията `xor` с 1, резултатът ще бъде 0. Ако желания бит е имал стойност 0, резултатът от приложената върху него операция `xor` с 1 ще бъде 1. Останалите битове ще запазят стойността си.

Задачи

Задача 2
Да се напише програма, която разменя стойностите на две променливи.

- Да се използва временна променлива. Защо операциите a = b; и b = a; не дават желания резултат?

```
int a, b; // въвеждаме стойностите от клавиатурата
int temp = a; a = b; b = temp;
```

- Без временна променлива

```
int a, b; // въвеждаме стойностите от клавиатурата
a = a + b; b = a - b; a = a - b;
```

Задача 3
Да се напише програма, която намира средно аритметичното на две цели числа.

Да се обърне внимание на операцията делене, която може да бъде целочислена, ако и двата й аргумента са цели числа!

```
int a, b; // въвеждаме стойностите от клавиатурата
double avg = (a + b)/2.0; // за да не се окаже целочислено делене

double avg = static_cast<double>(a + b)/2;
```

Преобразуване на типове? Дали не е рано още? Да се остави за следващото упражнение.

Задача 4
Да се напише програма, която прочита 3 цифри от стандартния вход и конструира трицифрено число от тях.

```
int number = (a*10 + b)*10 + c;
```

Задача 5
Да се напише програма, която прочира трицифрено цяло положително число от стандартния вход и извежда на екрана цифрите на единиците, десетиците и стотиците.

```
// цифра на единиците, десетиците и стотиците съответно
number % 10, (number / 10) % 10, number / 100
```

Задача 6
Да се напише програма, която намира по-малкото от две цели числа.

За да се използва функцията `abs(x)` трябва да се включи библиотеката `<cmath>`.

За изчислението може да се използва формулата: $\min(a,b) = \frac{a+b-|a-b|}{2}$

Задача 7
Да се напише програма, която прочита от клавиатурата три цели числа и извежда на екрана броя на положителните.

```
countPositive = (a > 0) + (b > 0) + (c > 0);
```

В контекст а аритметични оператори `True` се разглежда като `1`, а `False` като `0`.

Задача 8
Да се запише булев израз, който да има стойност истина, ако посоченото условие е вярно и стойност лъжа в противен случай.

a) цялото число е четно;
b) цялото число се дели на 4 или на 7.
c) уравнението $a.x^2 + b.x + c = 0, a \neq 0$ няма реални корени.
d) x принадлежи на интервала [0; 1];
e) x е извън интервала [0; 1];
f) цифрата 7 влиза в записа на трицифрено число;
g) поне две от цифрите на трицифрено число са равни помежду си.

Литература

Тодорова, М. (2010). *Програмиране на C++: част първа* (второ издание изд.). София: Сиела.

Тодорова, М., Армянов, П., Петкова, Д., & Георгиев, К. (2008). *Сборник от задачи по програмиране на C++: част първа Увод в програмирането*. София: ТехноЛогика.

Lippman, S., Lajoie, J., & Moo, B. (2007). *C++ Primer* (4th изд.). Addison-Wesley.

Prata, S. (2011). *C++ Primer Plus* (6th изд.). Addison-Wesley.