Въведение

- Водещи
- Цели на курса
- Протичане на занятия
- Домашни
- Контролни
- Присъствие

Основни понятия

- **Програма –** редица от инструкции, която решава определена задача

- **Програмиране –** създаване на редицата от инструкции

- **Алгоритъм –** механизъм за намиране на решение на поставена задача

- **Език за програмиране** - формални езици за описание на данните и алгоритмите и за тяхната обработка посредством компютри. Примери - Pascal, C++, C#, JavaScript,π Java, Basic, PHP, Python, Perl, Assembler, Logo, Ruby и много други

- **Език от ниско ниво –** Асемблер / -**Език от високо(средно) ниво** - C++

- **Видове транслатори**

интерпретатори – командите се превеждатπ една по една и непосредствено след това се изпълняват. Пример: VisualBasic, Python, PHP

компилатори – първо цялата програма сеπ превежда на машинен език, процеса се нарича компилиране. След това може да бъде изпълнена многократно, при голямо бързодействие.

- **какво е IDE?(integrated development environment)**

Набор от взаимодействащи си програми, събрани заедно и свързани една с друга с цел да обхванат всеки един етап от разработката на дадена програма.

Компоненти на една среда за програмиране

текстов редактор – за създаване и редактиране на програмния код

транслатор (компилатор) – проверява програмата за синтактични грешки и я превежда до обектен код

свързващ редактор (линкер) – свързва всички програмни файлове и използваните от тях библиотеки в изпълним (EXE) файл

дебъгер – за постъпково изпълнение на програмата и откриване на програмни грешки

- **синтаксис на C++(основни елементи, основни думи)**

Допустими в програмите са следните символи:

малки и главни латински букви

цифрите от 0 до 9

специални символи + - * / = () [] {} |θ & ! < > # $ ^ % ^ ~ _ . , : ; ‘ "

разделители – интервал, табулация и нов ред

Езикът прави разлика между малки и главни букви: ime ≠ Ime ≠ IME for ≠ For ≠ FOR

Думи на езика

запазени (ключови) думи – имат значение, което не може да бъде променяно. Обозначава езиковите конструкции. Например: if, for, while, break

стандартни думи – включени в стандартните библиотеки на езика. Имат специално значение, което обаче може да бъде предефинирано. Пример: cout, cin, sqrt, ceil

идентификатори – за именоване на създадени от потребителя променливи, константи, типове данни и др. Трябва да започват с буква или _ и да съдържат букви, цифри и _ Коректно: Min, MaxN, A5, _start, AA, B105 Погрешно: 10a, Sredna Suma, Nai.Malko, СУМА

- **коментари – едноредов и многоредов**

Части от кода на програмата, които се игнорират при изпълнението и. Служат за обяснение на програмата или за временно изключване на код, който искаме да не се изпълнява

// еднoредов коментар

/* многоредов коментар */

коментари от един тип не се влагат едни в други и не се слагат коментари за очевидни неща

Структура на програмата

#include <библиотеки>

// глобални декларации и функции

int main() {

// локални декларации

// команди

return 0;

}

- **променливи**

Място в паметта

Име (идентификатор)

Тип

Стойност

Деклариране на променлива

общ вид: тип име ; / тип име =стойност;

действие: декларира се променлива с указаните тип и име и евентуално се инициализира със стойност

пример: int a, b; double c=12.5;

Деклариране на константа

общ вид: const тип име =стойност;

действие: декларира се константа с указаните тип, име и стойност

пример: const double pi=3.14; предимства: лесна промяна на стойността на константата;

Команда за присвояване

общ вид: променлива=израз;

действие: изчислява се стойността на израза и той се присвоява на променливата пример: suma=2+3*4; a=b=c=0;

- cin/cout – вход/изход от конзолата

Команда за въвеждане

общ вид: cin>>променлива;

действие: изчаква се въвеждане на стойност от клавиатурата и тя се присвоява на променлива. Стойностите може да са разделени с интервал или знак за нов ред.

пример: cin>>n; cin>>a>>b>>c;

особености: cin е дефинирана в библиотеката iostream.h

Команда за извеждане

общ вид: cout<<израз;

Действие: изчислява се израза и неговата стойност се извежда на екрана. За добавяне на нов ред се използва endl.

Пример: cout<<2+2;   cout << a <<" + "<<b<<" = "<<a+b<<endl;

особености: cout е дефинирана в библиотеката iostream.h

- типове данни

Всяка променлива, всеки израз в C++ има тип. Типът определя

Множество от допустимите стойности

Операции

Когато декларираме променлива, трябва да определим нейния тип.

Целочислени типове

Предимно използваме int

Множество от стойности: [-231; 231-1] (4 байта)

Модификатори: short [-215; 215-1] long [-263; 263-1] unsigned [0; 2^x -1] (x = 16, 32, 64)

Операции: +a, -a (унарни операции) , +, -, * (умножение), / (частно), % (остатък)

Числа с плаваща запетая (floating point)

double (8 байта), float (4 байта)

Примери: 1, 2.34, 12e-2, 10.14E+03, .23

Операции: всички за целочислен тип без %

Вградени математически функции -> #include <cmath/math.h>

abs(x) – абсолютна стойност, x е целочислен израз,

fabs(x) – абс. стойност, x е floating point

sin(x), cos(x), tan(x), asin(x), acos(x), atan(x)

exp(x), log(x) – натурален, log10(x)

ceil(x) – закръгляне нагоре, floor(x) – надолу

sqrt(x) – корен квадратен, pow(x, n) – x^n

- операции

Аритметични операции

унарен плюс: +

унарен минус: -

събиране: +

изваждане: -

умножение: *

деление: /

остатък: %

Приоритет на операциите

изразите в скоби: ( )

унарни операции: + -

умножение, деление: * / %

събиране, изваждане: + -

Операции за сравнение

по-малко: <

по-малко или равно: <=

по-голямо: >

по-голямо или равно: >=

равно: ==

различно: !=

- GitHub и git контрол и Elms.fmi.uni-sofia.bg – безплатен софтуер за студенти на ФМИ

- интересни задачи

1.Лице и обем на пирамида

Да се намери лице и обем на правилна пирамида с основа квадрат. Страната на квадрата е 4. Апотемата е 6. Височината на пирамидата е 5.

V = 1/3*S/h (където S е лицето на основата, h –височината на пирамидата)

S1 = P.k/2 + S (P е периметъра на основата, к – апотема, S лице на основата)

2. Сума от цифрите на число

Да се въведе от клавиатурата положително 3-цифрено цяло число. Да се намерят и отпечатат цифрите му и да се намери сумата им.

s = a/ 100 d = a/10%10 e = a%10

Примерни решения:

```
#include <iostream>


using namespace std;


int main()
{
   // first task
   cout << "Hello world!" << endl;
   cout << "o   o ooooo o   o   ooooo   o       o ooooo ooooo o   oooo  o" << endl;
   cout << "o   o o   o   o   o o   o       o o  o o  o o   o  o  o" << endl;
   cout << "oooooo ooooo o   o   o o   o  o  o o  o ooo  o   o  o o" << endl;
   cout << "o   o o   o   o   o o   oo  oo  o o o  o o   o  o  " << endl;
   cout << "o   o ooooo ooooo ooooo  ooooo     oo  oo   ooooo o  o ooooo oooo  o" << endl;


   // second task
   double a = 4;
   double k = 6;
   double h = 5;


   cout << "S = " << (4*a*k)/2 + a*a << endl;
   cout << "V = " << (a*a*h)/3 << endl;
```

```
    // third task

    int number;
    cin >> number;

    int s = number / 100;
    int d = number / 10 % 10;
    int e = number % 10;

    cout << s + d + e << endl;

    return 0;
}
```

Какво е git и за какво ще го използваме в курса?

Git е контрол на версиите.

Ако питате чичко гоогле ще ви каже, че: "Система за контрол на версиите (на английски: Version control system) е механизмът, по който се управлява работата по даден софтуерен проект. За да се улесни разработката на софтуер са създадени специални системи, които намаляват неудобствата при съвместна работа на много хора върху един проект."

За качване в GitHub може да използвате две възможностти чрез терминал или чрез приложението от GitHub

Част от командите в GitHub:

1) git clone "URL"

- URL е много препорачително (направо задължително да е SSH);
- може да го намерите в дясната част на екрана;
- това сваля всичко, което е качено в GitHub във ваше локално копие (нужно ви е да го правите само 1 път, ако стоите на едно PC)

2) git status

- показва ни дали нашия проект има някакви промени (да трябва да качим нещо, че нещо е било променено и още такива интересни работи);
- препоръчително е тази команда да се изпълнява след всяка друга команда(да знаем дали всичко е минало както трябва);

3) git add -A || git add main.cpp Person.h .....

- git add -A добавя всички промени;
- git add main.cpp .... добавя само изредените след add файлове;

4) git commit -m"We love c++"

- качва нещата в ЛОКАЛНОТО репо

- в кавичките след m трябва да напишем съобщение, което описва направените от нас промени(не е хубаво да пишем романи)

Тук е мястото да кажем, че имаме две нива съхранение.

- локално репо

- GitHub (нашето публично място, където се хвалим пред хората, че сме умни)

До момента ние правим промени единствено на нашата машина (нищо от това не се е отразило в публичното репо)

5) git push -u origin master

- тук вече всичко става публично

- преди да сме качили нещо трябва да сме сигурни, че сме свалили всичко от публичното репо(ако не е така то ще ни гръмне и ще разберем, че нещо не е като хората)

- master е едно много хубаво място, но за сега няма да обясним какво е то :)

6) git pull origin master

- с тази команда сваляме всичко, което имам в GitHub за това репо

- ако 5) изгърми значи сме забравили да изпълним тази команда

- хубаво е преди да качваме нещо, да изпълняваме тази команда (може другарчето да е качило нещо и ние да го нямаме)

Има още много работи да се говорят за тези системи, но за сега ще остане това.