


Увод в програмирането

**Основни елементи от
програмирането на C++**



C++

- **C++ е език за обектно-ориентирано програмиране.**
- Създаден е от Бярне Страуструп от AT&T Bell Laboratories в края на 1985 година
- C++ е разширение на езика C в следните три направления:
 - създаване и използване на абстрактни типове данни;
 - обектно-ориентирано програмиране;
 - подобрения на конструкции на езика C (производни типове, наследяване, полиморфизъм).

Пример за програма на C++

- **Задача 1.** Да се напише програма, която намира периметъра и лицето на правоъгълник със страни 2,3 и 3,7.

Пример за програма на C++ ...

```
// Program Zad1.cpp
#include <iostream.h>
int main()
{ double a = 2.3;
  double b = 3.7;
  double p, s;
  /* намиране на периметъра на правоъгълника */
  p = 2*(a+b);
  /* намиране на лицето на правоъгълника */
  s = a*b;
  /* извеждане на периметъра */
  cout << "p = " << p << "\n";
  /* извеждане на лицето */
  cout << "s = " << s << "\n";
  return 0;
}
```

Дефиниране на променливи

- С++ е строго типизиран език за програмиране. Всяка променлива има тип, който *явно* се указва при дефинирането ѝ.

- *Синтаксис*

<име_на_тип> <променлива> [= <израз>]
 {, <променлива> [= <израз>]};

Където

- <име_на_тип> е дума, означаваща име на тип като int, double и др.;
- <израз> е правило за получаване на стойност – цяла, реална, знакова и друг тип, съвместим с <име_на_тип>.

Дефиниране на променливи ...

- *Семантика*

Дефиницията свързва променливата с множеството от допустимите стойности на типа, от който е променливата или с конкретна стойност от това множество. За целта се отделя определено количество оперативна памет (толкова, колкото да се запише най-голямата константа от множеството от допустимите стойности на съответния тип) и се именува с името на променливата. Тази памет е с неопределена стойност или съдържа стойността на указания израз, ако е направена инициализация.

- *Пример:*

```
double a = 2.3;  
double b, p, s;
```

Коментар

- Коментарът е произволен текст, ограден със знаците `/*` и `*/` или от `//` и ENTER. Игнорира се напълно от компилатора.

- *Синтаксис*

`<коментар> ::= /* <редица_от_знаци> */ |
 // <редица_от_знаци> ENTER`

- *Семантика*

Пояснява програмен фрагмент. Предназначен е за програмиста. Игнорира се от компилатора на езика.

Оператор за присвояване

- *Синтаксис*

<променлива> = <израз>;

като <променлива> и <израз> са от един и същ тип.

- *Семантика*

Пресмята стойността на <израз> и я записва в паметта, именувана с променливата от лявата страна на знака за присвояване =.

- *Пример:*

p = 2 * (a+b) ;

Изпълнение на Zad1.cpp

```
// Program Zad1.cpp
#include <iostream.h>
int main()
{
    double a = 2.3;
    double b = 3.7;
    double p, s;
    /* намиране на периметъра на правоъгълника */
    p = 2*(a+b);
    /* намиране на лицето на правоъгълника */
    s = a*b;
    /* извеждане на периметъра */
    cout << "p = " << p << "\n";
    /* извеждане на лицето */
    cout << "s = " << s << "\n";
    return 0;
}
```

ОП:

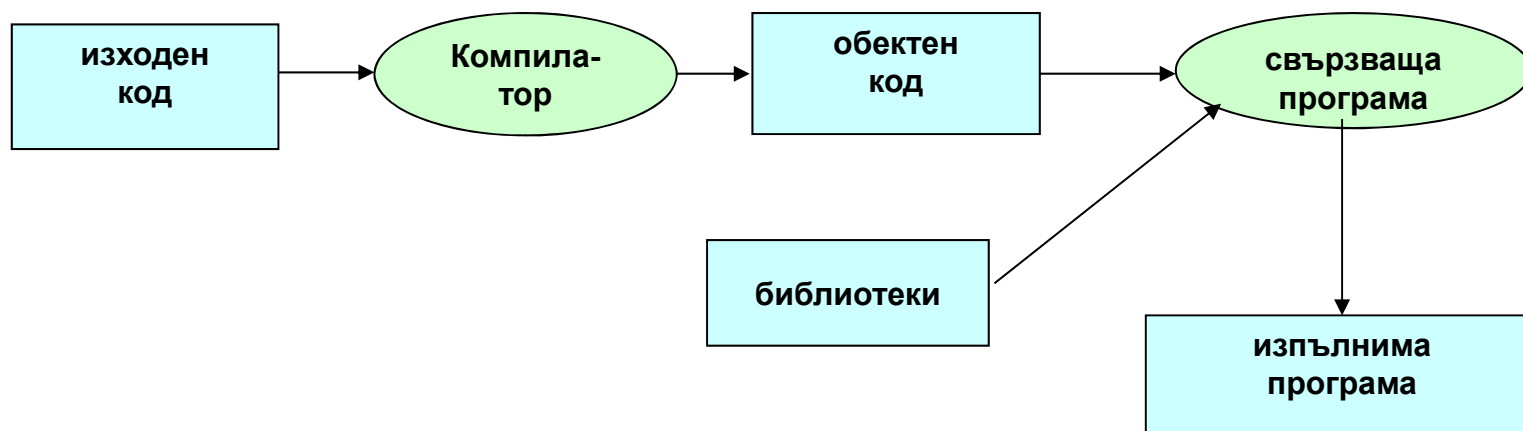
a	b	p	s
2.3	3.7	-	-

a	b	p	s
2.3	3.7	12.0	8.51

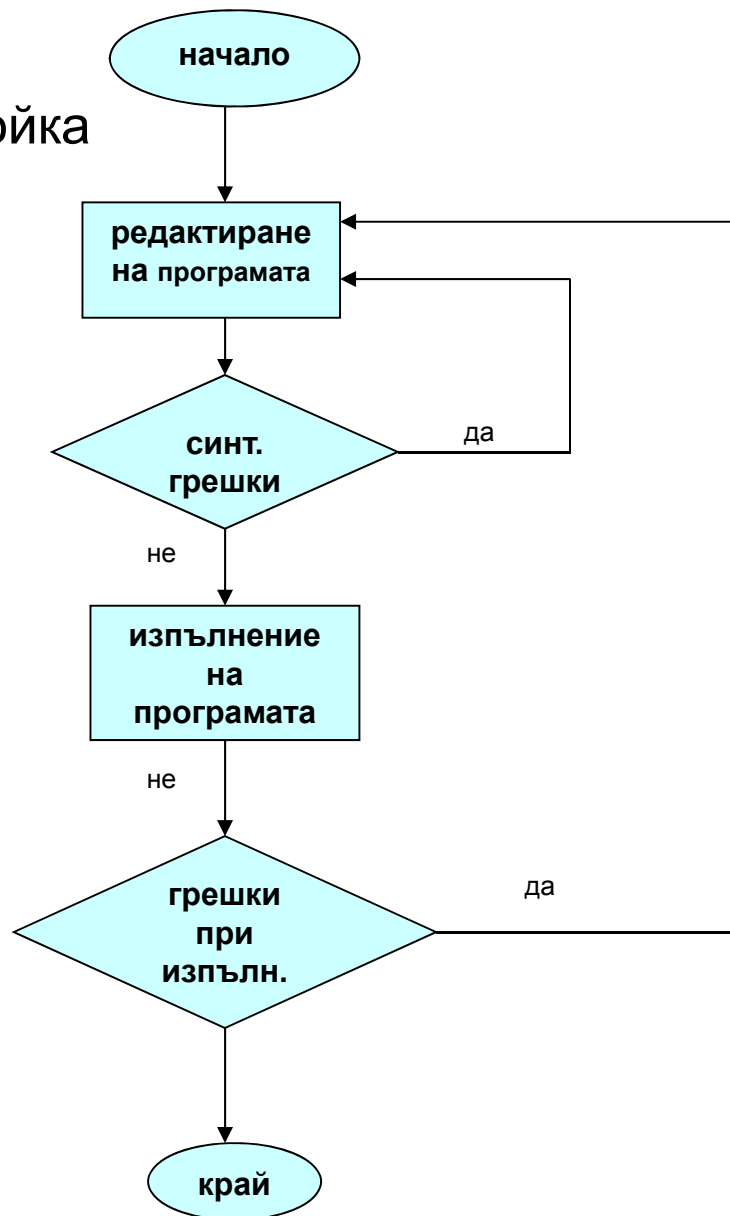
Изпълнение на програма на езика C++

- Стъпки:
 1. Създаване на изходен код
Zad1.cpp
 2. Компилиране
обектен код
Zad1.obj
 3. Свързване
изпълним код
Zad1.exe

Изпълнение на програма на езика C++ ...



Цикъл редактиране – компилиране - настройка



Азбука на езика C++

- главните и малки букви на латинската азбука;
- цифрите;
- специалните символи

+ - * / = () [] { } | : ; “ ‘ < > , .
_ ! @ # \$ % ^ ~



Думи на езика

- Думите на езика са:

- ☐ идентификатори,
- ☐ запазени и стандартни думи,
- ☐ константи,
- ☐ оператори
- ☐ и препинателни знаци

Идентификатори

- Редица от букви, цифри и знака за подчертаване (долна черта), започваща с буква или знака за подчертаване, се нарича **идентификатор**

- *Примери:*

A12 help Help double int15_12
rat_number INT1213 Int15_12

1ba ab+1 a(1) a'b

- $\langle \text{променлива} \rangle ::= \langle \text{идентификатор} \rangle$

Запазени думи

- Това са такива идентификатори, които се използват в програмите по стандартен, по предварително определен начин и които не могат да бъдат използвани по друг начин. Чрез тях се означават декларации, дефиниции, оператори, модификатори и други конструкции.
- В програмата Zad1.cpp са използвани запазените думи `int`, `double`, `return`

Стандартни думи

- Това са такива идентификатори, които се използват в програмите по стандартен, по предварително определен начин. Тези идентификатори **могат** да се използват и по други начини, например като обикновени идентификатори.

- Пример cout в Zad1.cpp

```
#include <iostream.h>
int main()
{ int cout = 21;
  return 0;
}
```

Константи

- Информационна единица, която не може да бъде променяна, се нарича **константа**. Има числови, знакови, низови и др. типове константи.
- Целите и реалните числа са **числови константи**.
- Целите числа се записват както в математиката и могат да бъдат задавани в десетична, шестнадесетична или осмична бройна система.

Константи ...

- Реалните числа се записват по два начина:
 - във формат с *фиксирана точка*
(например, 2.34 -12345.098)
 - и в *експоненциален формат*
(например, 5.23e-3 или 5.23E-3).
- **Низ, знаков низ или символен низ** е крайна редица от знаци, оградени в кавички.
Например:
"Това е низ.", "1+23-34", "Hello\n"

Оператори

- В C++ има три групи оператори:
 - *аритметично-логически оператори*
 - *управляващи оператори*
 - *операторите за управление на динамичната памет*

Препинателни знаци, разделяне на думите, коментари

- *Препинателни знаци*

Използват се ; < > { } () и др. знаци.

- *Разделяне на думите*

В C++ разделителите на думите са интервалът, вертикалната и хоризонталната табулации и знакът за нов ред.

- *Коментари*

Коментарите са текстове, които не се обработват от компилатора. В C++ има два начина за означаване на коментари. Единият начин е, текстът да се огради с /* и */. Другият начин са коментарите, които започват с // и завършват с края на текущия ред.



Вход и изход

- **Задача 2.** Да се напише програма, която въвежда размерите на правоъгълник и намира периметъра и лицето му.

```
// Program Zad2.cpp
#include <iostream.h>
int main()
{ // въвеждане на едната страна
  cout << "a= ";
  double a;
  cin >> a;
  // въвеждане на другата страна
  cout << "b= ";
  double b;
  cin >> b;
  // намиране на периметъра
  double p;
  p = 2*(a+b);
  // намиране на лицето
  double s;
  s = a*b;
  // извеждане на периметъра
  cout << "p= " << p << "\n";
  // извеждане на лицето
  cout << "s= " << s << "\n";
  return 0;
}
```

Оператор за вход >>

■ Синтаксис

`cin >> <променлива>;`

където

- `cin` е обект (променлива) от клас (тип) `istream`, свързан с клавиатурата,
- `<променлива>` е идентификатор, дефиниран, като променлива от “допустим тип”, преди оператора за въвеждане. (Типовете `int`, `long`, `double` са допустими).

■ Семантика

Извлича (въвежда) от `cin` (клавиатурата) поредната дума и я прехвърля в аргумента-приемник `<променлива>`

Оператор за вход >>

■ *Общ синтаксис*

cin >> <променлива> {>> <променлива>};

Изразът

cin >> <променлива₁> >> <променлива₂> >> ... >> <променлива_n>;

е еквивалентен на

cin >> <променлива₁>;

cin >> <променлива₂>;

...

cin >> <променлива_n>;

Оператор за вход >>

■ Примери

```
double a, b, c;  
cin >> a >> b >> c;
```

1.1 2.2 3.3 ENTER

или

1.1 ENTER

2.2 ENTER

3.7 ENTER

```
int a;  
cin >> a;
```

1.25 ENTER

Оператор за изход <<

■ Синтаксис

`cout << <израз>;`

където

- `cout` е обект (променлива) от клас (тип) `ostream`, свързан с екрана на компютъра,
- `<израз>` е израз от допустим тип. Представата за израз продължава да бъде тази от математиката. Допустими типове са `bool`, `int`, `short`, `long`, `double`, `float` и др.

■ Семантика

Операторът `<<` изпраща (извежда) към (върху) `cout` (екрана на компютъра) стойността на `<израз>`.

Оператор за изход <<

■ *Общ синтаксис*

```
cout << <израз> {<< <израз>};
```

Изразът

```
cout << <израз1> << <израз2> << ... << <изразn>;
```

е еквивалентен на

```
cout << <израз1>;
```

```
cout << <израз2>;
```

...

```
cout << <изразn>;
```

Структура на програмата на C++

- Изходните файлове се организират по следния начин:

<изходен_файл> ::=

<заглавен_блок_с_коментари>

<заглавни_файлове>

<константи>

<класове>

<глобални_променливи>

<функции>

Структура на програмата на C++

■ *Заглавен блок с коментари*

Всеки модул започва със заглавен блок с коментари, даващи информация за целта му, за използваните компилатор и операционна среда, за името на програмиста и датата на създаването на модула.

■ *Заглавни файлове*

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

```
#include <cmath.h>
```

Константи

- За да бъде програмата по-лесна за четене и модифициране, е полезно да се дават символични имена не само на променливите, а и на константите. Това става чрез дефинирането на константи.
- **Задача 3.** Да се напише програма, която въвежда радиуса на окръжност и намира и извежда дължината на окръжността и лицето на кръга с дадения радиус.

```
// Program Zad3.cpp
#include <iostream.h>
const double PI = 3.14159265;
int main()
{ double r;
  cout << "r = ";
  cin >> r;
  double p = 2 * PI * r;
  double s = PI * r * r;
  cout << "p = " << p << "\n";
  cout << "s = " << s << "\n";
  return 0;
}
```


Дефиниране на константи

■ Синтаксис

const <име_на_тип> <име_на_константа> = <израз>;

където

- const е запазена дума (съкращение от constant);
- <име_на_тип> е идентификатор, означаващ име на тип;
- <име_на_константа> е идентификатор <израз> е израз от тип, съвместим с <име_на_тип>.

■ Семантика

Свързва <име_на_константа> със стойността на <израз>. Правенето на опит да бъде променена стойността на константата предизвиква грешка.

Константи

■ Примери

```
const int MAXINT = 32767;
```

```
const double RI = 2.5 * MAXINT;
```

■ Предимства

- ☐ Програмите стават по-ясни и четливи.
- ☐ Лесно (само на едно място) се променят стойностите им (ако се налага).
- ☐ Вероятността за грешки, възможни при многократното изписване на стойността на константата, намалява.



Класове

- Тази част съдържа дефинициите на класовете, използвани в модула.
- В езика C++ има стандартен набор от типове данни като `int`, `double`, `float`, `char` и др. Този набор може да бъде разширен чрез дефинирането на класове.
- Дефинирането на клас въвежда нов тип, който може да бъде интегриран в езика. Класовете са в основата на обектно-ориетираното програмиране, за което е предназначен езикът C++.

Глобални променливи

- Езикът поддържа глобални променливи. Те са променливи, които се дефинират извън функциите и които са “видими” за всички функции, дефинирани след тях. Дефинират се както се дефинират другите (локалните) променливи.

Функции

- Всеки програма задължително съдържа функция `main`.
- Възможно е да съдържа и други функции. Тогава те се изброяват в тази част на модула. Ако функциите са подредени така, че всяка от тях е дефинирана преди да бъде извикана, тогава `main` трябва да бъде последна. В противен случай, в началото на тази част на модула, трябва да се **декларират** всички функции.