

Наръчник
Как се научих да програмирам на C++ с
помощта на изкуствен интелект

Шенер Юмер, 2401321044

Използван LLM: Aria

Съдържание

Предговор	3
1 Въведение	4
1.1 Първи стъпки в C++	5
1.1.1 Компиляция и изпълнение	5
1.1.2 Как да компилираме и изпълним програма?	5
1.1.3 Имплементиране на български език в C++	6
1.1.4 Първата ни програма	6
1.2 Ключови елементи на синтаксиса и семантиката на C++	7
1.2.1 Ключови думи	7
1.2.2 Специални символи	8
1.2.3 Идентификатори	10
1.2.4 Литерали	11
1.2.5 Променливи и константи	13
1.2.6 Подпрограми (Функции)	14
1.3 Стандартен конзолен I/O в C++	14
1.4 Структуриране на C++ програма	14
1.4.1 Директиви към предпроцесора	14
1.4.2 Области на имената	14
2 Типове данни	15
2.1 Типове данни за цели и реални числа	15
2.2 Типове данни за символ и низ	15
2.3 Логически данни	15
3 Конструкции за поточен контрол	16
3.1 Конструкции за разклонение	16
3.2 Конструкции за цикъл	16
3.3 Конструкции за прекъсване	16

<i>СЪДЪРЖАНИЕ</i>	2
4 Съставни типове данни	17
4.1 Масиви	17
4.2 Структури	17
4.3 Класове	17
4.4 Обединения	17
4.5 Изброяване	17
4.6 Колекции	17
5 Манипулиране на паметта	18
5.1 Указатели	18
5.2 Референции	18
5.3 Адресна аритметика	18
5.4 Динамично и статично разпределяне на паметта	18

Предговор

Здравей, скъпи читателю!

Добре дошъл в твоето пътуване през света на програмирането, пътуване, което започна с любопитство и ще завърши с осъзнаването, че дори изкуственият интелект може да бъде учител.

Тази книга е както проектна работа, така и плод на моя опит в изучаването на C++ , език, който е едновременно мощен и взискателен. С помощта на изкуствения интелект Agia от Orega GX, аз се научих да разбирам абстрактни концепции, да решавам сложни проблеми и да създавам код, който работи.

На тези страници ще ви покажа уроците, които научих от Agia, и стъпка по стъпка ще ви науча как и вие да овладеете тънкостите на езика C++ . Ще ви докажа, че чрез помощта на изкуственият интелект можете да извлечете изключително много информация за изучаването на програмни езици и по този начин да се вдъхновите да се впуснете в света на програмирането.

Нека това пътуване ви вдъхнови да преоткриете собствения си потенциал и да осъзнаете, че нищо не е невъзможно!

Глава 1

Въведение

Добре дошли в света на C++ ! Този език за програмиране е истински гигант, който стои в основата на безброй приложения, игри и технологии, които използвате всеки ден. C++ е език, който ви дава мощта да създавате сложни и ефективни програми, да контролирате хардуера на компютъра си и да реализирате най-смелите си идеи.

Но C++ не е за начинаещи. Той е мощен и гъвкав, но е и сложен и изисква задълбочено разбиране.

Какво прави C++ толкова специален?

- Обектно-ориентирано програмиране - C++ е език, който ви позволява да структурирате програмите си около обекти, които съдържат данни и функции. Това е като да създадете симулация на реалния свят в код, където всеки обект е отделен елемент с собствени характеристики и поведение.
- Висока производителност - C++ е известен с ефективността си. Той ви дава пълен контрол над ресурсите на компютъра и ви позволява да създавате приложения, които работят бързо и ефективно.
- Гъвкавост - C++ е гъвкав език, който ви позволява да разработвате разнообразни приложения, от операционни системи и игри до приложения за мобилни устройства.
- Широко разпространен - C++ е широко разпространен език, който се използва от милиони програмисти по целия свят. Това означава, че ще имате лесен достъп до ресурси, общности и поддръжка.

Защо да се учите на C++ ?

- Мощни приложения - C++ ви дава мощта да създавате комплексни и ефективни приложения, които могат да решават трудни задачи.

- Дълбоко разбиране - C++ ви учи да разбирате как работи компютърът и как да управлявате ресурсите му.
- Отворена врата към нови възможности - C++ е отворена врата към широк спектър от професионални възможности.

В тази книга ще ви запознаем с основите на C++ и ще ви покажем как да създавате свои собствени програми. Пригответе се за вълнуващо пътешествие в света на програмирането!

1.1 Първи стъпки в C++

В тази глава ще се запознаем с основите на C++ , като започнем с компилация, компилиране и изпълнение на програми.

1.1.1 Компилация и изпълнение

C++ е компилиран език. Това означава, че кодът, който пишете, трябва да бъде преведен на машинно разбираем език, преди да може да се изпълни.

Компилацията е процес, който превръща изходния код (текстовият файл, който вие пишете) в изпълним файл. Изпълним файл е файл, който може да се изпълни от компютъра.

За да компилирате и изпълните C++ програма, ще ви е необходим компилатор. Компилатор е програма, която превежда изходния код на C++ в изпълним файл.

1.1.2 Как да компилираме и изпълним програма?

Ето стъпките, които трябва да следвате, за да компилирате и изпълните C++ програма:

- Създайте нов текстов файл с разширение .cpp.
- Напишете C++ кода си в този файл.
- Отворете командния ред (или терминал) и отидете до директорията, където е вашият текстов файл.
- Въведете следната команда, за да компилирате програмата:
`g++ име_на_файла.cpp -o име_на_изпълним_файл`
- Въведете следната команда, за да изпълните програмата:
`./име_на_изпълним_файл`

Пример:

Ако вашият файл се казва `hello.cpp` и искате да създадете изпълним файл `hello`, тогава трябва да въведете следните команди:

```
1 g++ hello.cpp -o hello
2 ./hello
```

1.1.3 Имплементиране на български език в C++

За да използваме български език в кода на C++ и в конзолата, трябва първо да зададем локализацията на проекта:

```
1 #include <locale>
2
3 int main() {
4     setlocale(LC_ALL, "Bulgarian");
5     // Локализация на кирилицата в проекта
6
7     return 0;
8 }
```

`#include <locale>` - Тази линия включва библиотеката `locale`, която ни дава достъп до функции за локализация на езици.

`setlocale(LC_ALL, "Bulgarian");` - Стандартна функция за локализация на български език.

1.1.4 Първата ни програма

Ето пример за проста C++ програма, която извежда текст на екрана:

```
1 #include <iostream>
2 #include <locale>
3
4 int main() {
5     setlocale(LC_ALL, "Bulgarian");
6
7     std::cout << "Hello, world!" << std::endl;
8     return 0;
9 }
```

`#include <iostream>` - Тази линия включва библиотеката `iostream`, която ни дава достъп до функции за вход и изход.

`int main()` - Тази линия дефинира главната функция на програмата. Всички C++ програми трябва да имат главна функция.

`std::cout << "Hello, world!" << std::endl;` - Тази линия извежда текста "Hello, world!" на екрана.

`return 0;` - Тази линия завършва изпълнението на програмата.

1.2 Ключови елементи на синтаксиса и семантиката на C++

1.2.1 Ключови думи

Ключовите думи в C++ са резервирани думи, които имат специално значение за компилатора. Те не могат да се използват като имена на променливи, функции или други идентификатори.

Ето някои от най-важните ключови думи в C++ :

Ключова дума	Описание
int, float, double, char, bool, void, auto, const, constexpr, decltype	Типове данни
if, else, else if, switch, case, default, break, continue, goto	Условни оператори
for, while, do while, break, continue	Цикли
return, sizeof, new, delete, nullptr, this	Оператори
namespace, using, struct, class, enum, union, template, typename, friend, operator	Организъм на кода
public, private, protected, static, virtual, override, final, explicit	Модификатори за достъп
inline, extern, volatile, mutable, register	Модификатори за компилация
try, catch, throw, noexcept	Обработка на изключения

Ключовите думи се използват в C++ код, за да се определят типове данни, структури, класове, функции и други елементи на програмата.

Пример:

```

1 int main() {
2     int a = 10; // int е ключова дума за дефиниране на целочислена променлива
3     if (a > 5) { // if е ключова дума за условен оператор
4         std::cout << "a is greater than 5" << std::endl;
5     }

```



```
6   return 0; // return е ключова дума за връщане на статус
7 }
```

1.2.2 Специални символи

В C++ езикът, освен букви, цифри и ключови думи, има и специални символи, които имат специално значение за компилатора. Тези символи се използват за определяне на оператори, разделителни символи, коментари и други елементи на кода.

Оператори

Операторите са специални символи, които извършват операции върху операнди.

Пример:

Аритметика

+ (събиране): $a + b$
- (изваждане): $a - b$
* (умножение): $a * b$
/ (деление): a / b
% (остатък от деление): $a \% b$
++ (увеличаване): $a++$
-- (намаляване): $a--$

Условни

== (равенство): $a == b$
!= (неравенство): $a != b$
> (по-голямо): $a > b$
< (по-малко): $a < b$
>= (по-голямо или равно): $a >= b$
<= (по-малко или равно): $a <= b$
&& (логическо И): $a \&\& b$
|| (логическо ИЛИ): $a || b$
! (логическо НЕ): $!a$

?: (тернарен условен оператор): $a ? b : c$

Битова аритметика

& (битово И): $a \& b$

| (битово ИЛИ): $a | b$

^ (битово ИЗКЛ. ИЛИ): $a \wedge b$

~ (битово НЕ): $\sim a$

<< (ляво битово изместване): $a \ll b$

>> (дясно битово изместване): $a \gg b$

Разделителни символи

Разделителните символи се използват за разделяне на различни части на C++ кода.

Пример:

; (точка и запетая): `int a = 10;`

, (запетая): `int a = 10, b = 20;`

: (двоеточие): `switch (a) { case 1: ...; }`

:: (обхват на име): `std::cout`

. (член на клас): `a.b`

-> (член на указател): `a->b`

[] (индексиране): `a[b]`

() (извикване на функция): `a()`

{ } (блокове код): `{ ... }`

Коментари

Коментарите са текст, който се игнорира от компилатора. Те се използват за обяснение на кода, добавяне на документация или деактивиране на част от кода.

Пример:

- // (едноредов коментар):

```
1 // This is a single line comment.
```

- `/* ... */` (многоредов коментар):

```
1 /*  
2  * This is a multiline comment.  
3  * It can extend on multiple lines.  
4  */
```

1.2.3 Идентификатори

Идентификаторите в C++ са имена, които се използват за означаване на променливи, функции, класове, структури, изброявания, области на имена и други елементи на програмата.

Правила за идентификатори

Идентификаторите могат да се състоят от букви, цифри, подчертаване (`_`). Първият символ на идентификатора не може да бъде цифра. Ключовите думи не могат да се използват като идентификатори. Чувствителност към регистъра: `myVariable` и `MyVariable` са различни идентификатори.

Примери за идентификатори

- Променлива: `age`, `firstName`, `totalScore`
- Функция: `calculateArea`, `printMessage`, `sortArray`
- Клас: `Person`, `Car`, `Database`
- Структура: `Point`, `Date`, `Time`
- Изброяване: `Color`, `Status`, `Direction`
- Област на имена: `std`, `myNamespace`, `utils`

Препоръки за идентификатори

Използвайте описателни имена, които отразяват целта на идентификатора. Използвайте `camelCase` или `snake_case` за по-добра четимост. Избягвайте къси и неясни имена. Не използвайте резервирани думи като идентификатори.

Пример за код с идентификатори:

```
1 #include <iostream>
2
3 using namespace std; // Дефиниране на областта на имената "std"
4
5 int main() {
6     // Дефиниране на променлива с име "age"
7     int age = 25;
8
9     // Дефиниране на функция с име "printMessage"
10    void printMessage(string message) {
11        cout << message << endl;
12    }
13
14    // Извикване на функцията "printMessage"
15    printMessage("Hello, world!");
16
17    return 0;
18 }
```

1.2.4 Литерали

Литералите в C++ са константни стойности, които се използват за представяне на данни в програмата. Те са директни представяния на данни, които се компилират директно в код.

Видове литерали

C++ поддържа различни видове литерали, в зависимост от типа на данните:

- Числови литерали:

- Цялочислени литерали:

- Десетични: 10, 25, -15

- Осмоични: 012, 037 (започват с 0)

- Шестнадесетични: 0x1A, 0x2F (започват с 0x)

- Дробни литерали: 3.14, 1.5e-2 (експоненциална нотация)

- Символни литерали:

- Обикновени: 'a', 'B', '%'

- Escape последователности: `'\n'`, `'\t'`, `'\'`
- Текстови литерали:
 - Обикновени: `"Здравей, свят! "Hello, world!"`
 - Raw string литерали: `R"(C:\Users\MyUser\Documents)"` (за запазване на escape последователности)
- Булеви литерали: `true`, `false`
- Указателни литерали: `nullptr` (за празен указател)

Пример за код с литерали

```
1 #include <iostream>
2
3 using namespace std;
4
5 int main() {
6     // Цялочислени литерали
7     int age = 25;
8     int octalNumber = 012;
9     int hexNumber = 0x1A;
10
11     // Дробни литерали
12     double pi = 3.14;
13     double smallNumber = 1.5e-2;
14
15     // Символни литерали
16     char character = 'A';
17     char newline = '\n';
18
19     // Текстови литерали
20     string message = "Hello, world!";
21     string path = R"(C:\Users\MyUser\Documents)";
22
23     // Булеви литерали
24     bool isTrue = true;
25     bool isFalse = false;
26
27     // Указателни литерали
28     int* ptr = nullptr;
29
30     return 0;
31 }
```

1.2.5 Променливи и константи

Променливи

Променливите в C++ са имена, които се използват за съхраняване на данни в паметта. Тези данни могат да бъдат променяни по време на изпълнението на програмата.

1. Дефиниране на променливи:

```
1 <datatype> <variable_name>;
```

Пример:

```
1 int age; // Дефиниране на променлива от тип "int" с име "age"
2 double price; // Дефиниране на променлива от тип "double" с име "price"
3 string name; // Дефиниране на променлива от тип "string" с име "name"
```

2. Инициализиране на променливи

Инициализирането на променлива е процесът на присвояване на начална стойност при дефинирането.

Пример:

```
1 int age = 25; // Инициализиране на променливата "age" със стойност 25
2 // Инициализиране на променливата "price" със стойност 19.99
3 double price = 19.99;
4 // Инициализиране на променливата "name" със стойност "Ivan"
5 string name = "Ivan";
```

3. Използване на променливи

След дефинирането и инициализирането, променливите могат да се използват в програмата.

Пример:

```
1 int age = 25;
2 // Извеждане на стойността на променливата "age"
3 cout << "Your age is: " << age << endl;
```

Константи

Константите в C++ са имена, които се използват за съхраняване на данни в паметта, но стойностите им не могат да се променят по време на изпълнението на програмата.

1. Дефиниране на константи

За да се дефинира константа, се използва ключовата дума `const`:

```
1 const <datatype> <NAME_OF_CONSTANT> = <value>;
```

Пример:

```
1 // Дефиниране на константа от тип "int" с име "MAX_AGE" със стойност 120
2 const int MAX_AGE = 120;
3 // Дефиниране на константа от тип "double" с име "PI" със стойност 3.14159
4 const double PI = 3.14159;
5 // Дефиниране на константа от тип "string" с име "GREETING"
6 // със стойност "Greetings!"
7 const string GREETING = "Greetings!";
```

2. Използване на константи

Константите могат да се използват в програмата по същия начин като променливите.

Пример:

```
1 const int MAX_AGE = 120;
2 int age = 25;
3 if (age > MAX_AGE) {
4     cout << "Invalid age!" << endl;
5 }
```

1.2.6 Подпрограми (Функции)

type, void, recursion, lambda

1.3 Стандартен конзолен I/O в C++

1.4 Структуриране на C++ програма

1.4.1 Директиви към предпроцесора

1.4.2 Области на имената

Глава 2

Типове данни

2.1 Типове данни за цели и реални числа

2.2 Типове данни за символ и низ

2.3 Логически данни

Глава 3

Конструкции за поточен контрол

3.1 Конструкции за разклонение

!! Ternary operator

3.2 Конструкции за цикъл

3.3 Конструкции за прекъсване

Глава 4

Съставни типове данни

4.1 Масиви

4.2 Структури

4.3 Класове

4.4 Обединения

4.5 Изброяване

4.6 Колекции

Глава 5

Манипулиране на паметта

5.1 Указатели

5.2 Референции

5.3 Адресна аритметика

5.4 Динамично и статично разпределяне на паметта