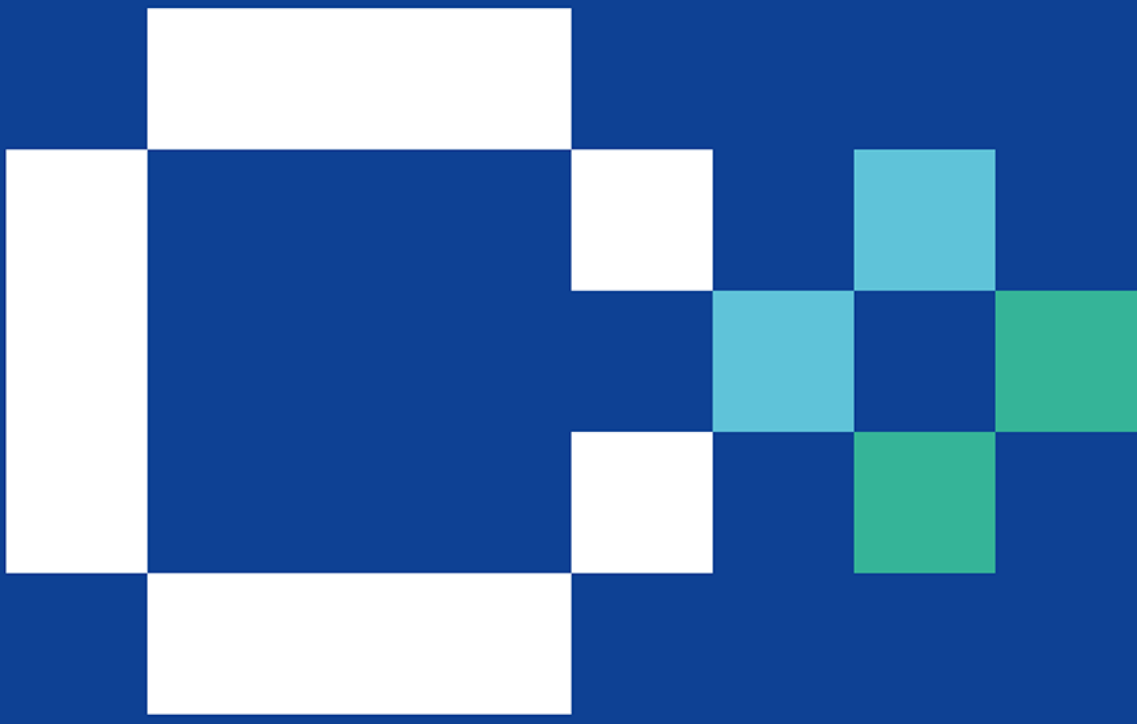
Maciej Matyka

Kombinacje C++



648 łamigłówek programistycznych z odpowiedziami

Helion

Redaktor prowadzący: Małgorzata Kulik

Projekt okładki: Studio Gravite/Olsztyn
Obarek, Pokoński, Pazdrijowski, Zaprucki

Materiały graficzne na okładce zostały wykorzystane za zgodą Shutterstock.

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *https://helion.pl* (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
https://helion.pl/user/opinie/komcpp
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

ISBN: 978-83-8322-310-0

Printed in Poland.

Spis treści

Wstęp

Idea tej książki przyszła mi do głowy niespodziewanie pewnego wiosennego, niedzielnego poranka. To taka pora, kiedy wpadamy na najlepsze pomysły. Moje myśli zeszły na szachy, bo za dwa tygodnie miałem grać z dziećmi w lokalnym turnieju o kategorie szachowe. Jestem szachistą amatorem. Lubię grać, ale nigdy nie studiowałem tego tematu, raczej grałem sobie dla zabawy — najpierw z tatą, potem z kurnikiem, a dzisiaj ze swoimi dziećmi. Przypomniałem sobie o książce, którą zacząłem przerabiać i w której byłem już na 7. (siódmym!) zadaniu z chyba 600, które tam opisano. Jest to książka *Kombinacje Szachowe* Jacka Gajewskiego i Jerzego Konikowskiego wydana w 2020 roku przez wydawnictwo RM. Wiedziałem, że ich przerobienie zajmie mi kilka dobrych miesięcy, bo są dla mnie bardzo trudne. Sam autor pisze, że każde z nich to około 20 minut pracy i nie należy książki czytać zbyt szybko, bo nie o to w niej chodzi. Pomyślałem, że gdyby ktoś pokazał mi te zadania szachowe wcześniej, może brałbym udział w jakichś turniejach i trochę mocniej wsiąknął w to środowisko.

Szachy to piękna gra. Kombinacje szachowe są nietrywialne i ich wykonywanie daje dużo frajdy. Przyszło mi do głowy, że magia tego zajęcia jest w pewien sposób interesująca. To w sumie niesamowite, że tak abstrakcyjna rzecz, tak na pozór absurdalne zajęcie jak rozwiązywanie kombinacji szachowych (i w dodatku tak trudne — ja niekiedy do jednego najprostszego diagramu podchodzę po kilka razy) może mnie jednak utrzymywać w pewnym napięciu i przyciągać.

I wtedy mnie olśniło. Przecież podobne diagramy i swego rodzaju łamigłówki można przełożyć na inne zagadnienie. Można stworzyć zadania, w których czytający musi się domyślić, o co chodzi, przeanalizować coś, kawałek odgadnąć, trochę pokombinować i znaleźć rozwiązanie. Jedna z podstawowych umiejętności programistycznych, jaką nabyłem na samym początku swojej zabawy z programowaniem, to umiejętność analizy prostych programów pisanych przez innych programistów. Pierwszy program zobaczyłem wydrukowany na kartce przez kolegę z klasy. Był napisany w języku Basic Commodore64. Pamiętam uczucie olśnienia, gdy zrozumiałem, co program ma zrobić, po prostu na niego patrząc.

Kiedy zaczynałem przygodę z programowaniem, nie było jeszcze internetu. Rolę informacyjną pełniły gazety drukowane na papierze. W nich często zamieszczano przedruki programów. Aby się nimi nacieszyć, trzeba było przepisać je ręcznie do komputera. Przypominam sobie, że mój leciwy VIC-20 nie miał możliwości ich prostego zapisu (albo nie umiałem tego robić, bo wtedy zapisywało się wszystko na kasetach) — chcąc trochę poprogramować, za każdym razem musiałem zaczynać przepisywanie od początku.

A teraz zawodowo zajmuję się dydaktyką programowania, uczę studentów podstaw języków C/C++ i próbuję od strony praktycznej pokazywać im, jak zapisać proste programy. Pomyślałem jednak, że uczę w sumie w standardowy sposób — nowy temat, najpierw teoria, kilka drobnych przykładów, potem rozwinięcie teorii, szczegóły i kolejny temat. Tymczasem wpadł mi do głowy pomysł: a może połączyć przerabiane przeze mnie diagramy z językiem programowania? Co, gdyby początkującemu programiście nie dawać od razu odpowiedzi, ale zaciekawić go i zainspirować prostą łamigłówką, problemem zapisanym w postaci programu? Języki programowania są dość proste, ich forma to takie trochę dłuższe zdania, czasem zapisane skrótowo, które wyrażają myśli programisty. Czytanie i analiza programów to podstawowe umiejętności, które można wyćwiczyć.

Czytelniku, oddaję Ci do rąk zestaw kilkuset diagramów programistycznych. Ich rozwiązanie powinno dać Ci podstawy do samodzielnego programowania w języku C++.

Układ książki

W pierwszej części omawiam podstawowe motywy programistyczne — wypisywanie danych do konsoli, zmienne, pętle, instrukcję warunkową, struktury, klasy i bibliotekę standardową. Początkujący mogą mieć z nimi problemy, więc zapraszam ich do przeczytania niewielkiego opisu tych motywów. Umieszczone w tej części diagramy zostały podzielone tematycznie. Dla doświadczonych programistów ich przerobienie może być dość proste — ci znajdą coś dla siebie dalej.

W drugiej części motywy będą już wymieszane, a zadania bardziej wymagające. Część z nich należy rozpisać na kartce (np. wypisać, co program po kolei robi, w pewnym sensie „wykonać" go ręcznie). Doświadczeni programiści potrafią to już zrobić w głowie i mam nadzieję, że właśnie do tego momentu dotrzemy wspólnie w tej książce. „Wykonywanie" programów w głowie to sztuka podobna do rozwiązywania diagramów szachowych, tylko zamiast skoczków i hetmana dysponujemy funkcjami i klasami, a zamiast szachownicy z 2^{64} kombinacji mamy kilka gigabajtów pamięci RAM.

Praca z książką

Nie należy się frustrować, jeśli nie umie się rozwiązać danego problemu. Ważniejsze od rozwiązania jest zrozumienie danego zagadnienia. Jeśli przyjdzie ono po dłuższej analizie, a potem po sprawdzeniu odpowiedzi i komentarza do zadania, to też należy sobie przyznać punkty. Uważam, że wartościowe jest zrozumienie zadania, a nie samo jego rozwiązanie. Mała wskazówka: warto patrzeć na wcześniejsze i późniejsze zadania i próbować wydedukować rozwiązanie. Umówmy się też, że na jedno zadanie nie poświęcamy więcej niż 20 minut. Myślę, że tempo jedno zadanie na dzień jest odpowiednie dla początkujących.

Jak pracować z zadaniami? Polecam w pierwszej kolejności spróbować „skompilować program w głowie", czyli zinterpretować, co on ma robić. Jeśli brakuje Ci wiedzy, poszukaj w książce podobnego zadania, wróć do wstępu, a może po prostu przepisz kod do kompilatora i go uruchom. Nie zaszkodzi też zajrzeć do klucza i po prostu przeanalizować, czemu wynik ma być taki, a nie inny. Niekiedy miks tych wszystkich rzeczy da to, co najważniejsze — pozwoli zrozumieć pewną technikę albo ideę programistyczną. Zachęcam, żeby się nie ograniczać i nie próbować stać przed murem. Jeśli nie da się programu zrozumieć, szukaj odpowiedzi w kluczu. Jeżeli to nie pomaga, przepisz go, uruchom i pobaw się nim. Jeśli to również nie przyniesie rezultatu, spróbuj znaleźć kogoś, kto zna się na rzeczy — na pewno chętnie Ci wszystko wytłumaczy.

Pamiętam, że ja swojego pierwszego programu nie umiałem zrozumieć na pierwszy rzut oka, a pokazał mi go chyba w 8 klasie szkoły podstawowej kolega ze szkolnej ławki (pozdrowienia dla Sebastiana). Musiałem ten program najpierw przepisać, uruchomić i następnie trochę w nim pozmieniać. I to mnie przyciągnęło do programowania — magia tego, że coś z kartki może wykonać pewne operacje i dać chwilę rozrywki, była niesamowita. Tak też można pracować z tą książką. Przepisanie programu jest w pewnym sensie podobne do jego rozwiązania — bo przypominam, że kluczowe jest pełne zrozumienie, jak program działa, a nie metoda dojścia do tego punktu. Umiejętność zrozumienia kodu bez jego przepisywania przyjdzie z czasem.

Dla kogo jest ta książka?

Książka jest dla każdego. Mogą z niej skorzystać zarówno początkujący chcący spróbować sił w programowaniu, jak i profesjonaliści, którzy pragną się pochwalić, że umieją rozwiązać naprawdę trudne diagramy z końca listy. Nauczyciele i wykładowcy mogą w niej szukać inspiracji do zajęć z programowania (zachęcam, żeby wykorzystywać diagramy jako przerywnik w trakcie lekcji czy wykładów). Studentom pomoże podnieść umiejętności. Przede wszystkim zaś, drogi Czytelniku, chciałbym zaprosić Cię do wspólnej zabawy.

Zadania

I. Wypisywanie tekstu

1. Co wypisze ten program?

```
#include <iostream>
int main(void)
{
   std::cout<<"Witaj świecie
   ↪C++"<<std::endl;
   return 0;
}
```

2. Co wypisze ten program?

```
#include <iostream>
using namespace std;
int main(void)
{
   cout << "Witaj świecie C++" << endl;
   return 0;
}
```

3. Co wypisze ten program?

```
#include <cstdlib>
#include <cstdio>

int main(void)
{
   printf("Język C na %d \n", 5);
   return 0;
}
```

4. Co wypisze ten program?

```
#include <iostream>

int main(void)
{
   std::cout << "T" << "a" << "k";
   std::cout << " " << "t" << "e";
   std::cout << " " << "z" << " " << "m";
   std::cout << "o" << "z" << "n" << "a";
   std::cout << std::endl;
   return 0;
}
```

5. Co wypisze ten program?

```
#include <iostream>

int main(void)
{
   std::cout << "2+3" << std::endl;
   return 0;
}
```

6. Co wypisze ten program? Porównaj z 5.

```
#include <iostream>

int main(void)
{
   std::cout << 2+3 << std::endl;
   return 0;
}
```

II. Zmienne

7. Co wypisze ten program?

```
#include <iostream>

int main(void)
{
  int a = 10;
  std::cout << a << std::endl;
  return 0;
}
```

8. Co wypisze ten program?

```
#include <iostream>

int main(void)
{
  int a = 10, b=20;
  std::cout << a+b << std::endl;
  return 0;
}
```

9. Co wypisze ten program?

```
#include <iostream>

int main(void)
{
  float x = 3.0;
  std::cout << x+0.14 << std::endl;
  return 0;

}
```

10. Co wypisze ten program?

```
#include <iostream>

int main(void)
{
  int a(4), b(5);
  int c(a+b), d=a-b;
  std::cout << c << " " << d <<
  ↪std::endl;
  return 0;
}
```

11. Co wypisze ten program?

```
#include <iostream>

int main(void)
{
  int a = 3;
  std::cout << 1/a << std::endl;
  return 0;
}
```

12. Co wypisze ten program?

```
#include <iostream>

int main(void)
{
  float x = 3;
  std::cout << 1/x << std::endl;
  return 0;
}
```

III. Wyszukiwanie błędów

13. Jaki błąd jest w tym programie?

```
include <iostream>
int main(void)
{
  std::cout<<"Witaj C++"<<std::endl;
  return 0;
}
```

14. Jaki błąd jest w tym programie?

```
#include <iostream>

int main(void)
{
  std::cout<<"Witaj C++"<<std::endl;
}
```

Klucz

1. `Witaj świecie C++` — `std::cout` jest tu obiektem, do którego przekazujemy to, co chcemy wypisać, a `std::endl` oznacza złamanie linii po wypisaniu tekstu.

2. `Witaj świecie C++` — `Using namespace std` pozwala pominąć ozdobniki `std::` i uprościć kod.

3. `Język C na 5` — użyta funkcja `printf` znana jest jeszcze z czasów języka C; oprócz wypisania tekstu wplatamy w niego też liczbę (tu 5) poprzez użycie wzorca `%d`, wskazującego, gdzie liczba będzie wpisana.

4. `Tak też można` — tekst jest wypisywany literka po literce.

5. `2+3` — działanie zostanie wypisane tekstem, bo jest umieszczone w cudzysłowie.

6. `5`, bo działanie jest tu elementem kodu i preprocesor najpierw je wyliczy i potem tu wstawi.

7. `10` — po przypisaniu wartości 10 do zmiennej `a` zostaje ona wypisana przez obiekt `std::cout` z użyciem operatora `<<`.

8. `30` — w programie najpierw policzona zostanie suma zmiennych `a` i `b`, a dopiero ich wynik przekazany zostanie do `std::cout`.

9. `3.14` — do wartości zmiennej `x` (równej 3.0) zostanie dodana wartość 0.14 i suma przesłana będzie do `std::cout`.

10. `9 -1` — podanie liczby w nawiasie pozwala też na przypisanie zmiennej jakiejś wartości, tu `a` = 4 oraz `b` = 5, zapisane jako `a(4)` i `b(5)`. Reszta jest jak wcześniej: zmienna `c` przyjmuje wartość sumy, a zmienna `d` wartość różnicy liczb `a` oraz `b`.

11. `0` — niespodzianka? Niekoniecznie. Komputer próbuje tu podzielić liczbę 1 przez 3 w trybie całkowitoliczbowym. To dlatego, że przypisaliśmy zmiennej `a` typ całkowity.

12. `0.333` — tu nie ma niespodzianki, a wyrażenie `1` podzielone przez zmiennoprzecinkowe `x` zostanie policzone, jak należy.

13. Brakuje znaczka `#` przed dyrektywą preprocesora dołączającą plik nagłówkowy `iostream`.

14. Brakuje rozkazu `return`. Funkcja `main` jest zadeklarowana jako zwracająca liczbę `int`, musi więc zawierać się w niej rozkaz `return` z parametrem — liczbą całkowitą.

Od dekad C++ pozostaje jednym

z dwóch najważniejszych języków, jeśli chodzi o programowanie systemowe, ale też aplikacyjne. Bez niego nie byłoby Windowsa, Excela, Photoshopa ani dużej części oprogramowania bazodanowego na czele z MySQL. A bez nich trudno sobie wyobrazić funkcjonowanie dzisiejszego świata. Nic więc dziwnego, że C++ od lat pozostaje obiektem zainteresowania programistów — w branży IT niezmiennie utrzymuje się silna potrzeba, aby doskonalić umiejętności programowania w tym języku.

Z pomocą przychodzi ta książka

— *Kombinacje C++. 648 łamigłówek programistycznych z odpowiedziami* to zbiór blisko 650 praktycznych zadań doskonalących znajomość języka C++ i umiejętności programistyczne. Układem nawiązuje do podręczników szachowych, które na konkretnych przykładach przybliżają obowiązujące zasady. Zawartość została podzielona na dwie części. Pierwsza zapoznaje z podstawowymi motywami, jak klasy, funkcje czy biblioteki standardowe. Druga część zawiesza poprzeczkę wyżej i wymaga więcej wysiłku — tym bardziej jednak zachęca do zaangażowania się w rozwiązywanie problemów, co w praktyce wprost przełoży się na podniesienie kompetencji programistycznych.

Dzięki książce:

- **zrozumiesz działanie programów**
- **nauczysz się wyszukiwać błędy w kodzie**
- **poznasz kluczowe motywy języka C++**
- **wyniesiesz swoje umiejętności na nowy, wyższy poziom**

Setki łamigłówek czekają!

Helion

helion.pl

HELION SA
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

KOD KORZYŚCI
Sięgnij po więcej! ▶

ISBN 978-83-8322-310-0

9 788383 223100

Cena: 57,00 zł