

Modularyzacja. Tablice dynamiczne

Wstęp do Informatyki i Programowania

Maciek Gębala

30 października 2025

Maciek Gębala Modularyzacja. Tablice dynamiczne

Modularyzacja

Podział kodu na osobne fragmenty mające logiczne uzasadnienie w działaniu (tworzenie bibliotek, pakietów)

Ułatwia pracę z kodem i ponowne użycie fragmentów oprogramowania.

Maciek Gębala Modularyzacja. Tablice dynamiczne

Modularyzacja w C

Plik nagłówkowy (końcówka .h) zawiera deklaracje funkcji do użycia. Odpowiadający mu plik z końcówką .c zawiera implementacje tych funkcji (z funkcjami pomocniczymi).

#include "plik.h" powoduje w czasie kompilacji wklejenie deklaracji funkcji z pliku plik.h i sprawdzenie czy są dobrze użyte. Połączenie skompilowanych plików następuje później.

Trzeba uważać aby kolejne wczytania pliku nagłówkowego nie spowodowały podwójnej deklaracji - #pragma once.

Maciek Gębala Modularyzacja. Tablice dynamiczne

Przykład w języku C

```
1 test.h
2 #pragma once
3 void jawne();
```

```
1 test.c
2 #include <stdio.h>
3 void tajne() { printf("Tajne\n"); }
4 void jawne() { printf("Jawne\n"); tajne(); }
```

```
1 main.c
2 #include "test.h"
3 int main() {
4     jawne();
5     tajne();
6
7     return 0;
8 }
```

Maciek Gębala Modularyzacja. Tablice dynamiczne

Notatki

Notatki

Notatki

Notatki

Modularyzacja z Adzie

Deklaracja pakietu funkcji/procedur zawarta jest w pliku z końcówką .ads i korzysta ze słowa kluczowego package. W deklaracji można wskazać części prywatne (niewidoczne przy korzystaniu z pakietu).

Implementacja jest w pliku z końcówką .adb i korzysta ze słów kluczowych package body.

Korzystanie z pakietu deklarujemy używając słowa kluczowego `with`. Słowo kluczowe `use` umożliwia pominięcie nazwy pakietu dla komponentu.

Maciek Gębala Modularyzacja. Tablice dynamiczne

Przykład w języku Ada

```
test.ads
package Test is
  procedure Jawne;
private
  procedure Tajne;
end Test;
```

```
test.adb
with Ada.Text_IO;
use Ada.Text_IO;

package body Test is
  procedure Jawne is
  begin
    Put_Line ("Jawne");
    Tajne;
  end Jawne;
  procedure Tajne is
  begin
    Put_Line ("Tajne");
  end Tajne;
end Test;
```

Maciek Gębala Modularyzacja. Tablice dynamiczne

Przykład w języku Ada

```
main.adb
with Test;

procedure Main is
begin
    Test.Jawne;
    -- Test.Tajne;
end Main;
```

```
main1.adb
with Test; use Test;

procedure Main1 is
begin
  Javne;
  -- Tajne;
end Main1;
```

Maciek Gębala | Modularyzacja. Tablice dynamiczne

Modularyzacja w Pythonie

Implementacja w osobnym pliku tworzy moduł o nazwie takiej jak plik. Umieszczenie pliku w podkatalogu powoduje dodanie nazwy katalogu do nazwy modułu.

Moduł dodajemy słowem kluczowym `import` - różne odmiany dodawania w przykładach.

Maciek Gębala Modularyzacja. Tablice dynamiczne

Notatki

Notatki

Notatki

Notatki

Przykład w języku Python

```
test.py
1 def jawne() :
2     print("Jawne")
3     tajne()
4
5 def tajne() :
6     print("Tajne")
```

```
main.py
1 import test
2
3 def main() :
4     test.jawne()
5     test.tajne()
6
7 if __name__ == "__main__" :
8     main()
```

Maciek Gębala

Modularyzacja, Tablice dynamiczne

Notatki

Przykład w języku Python

```
main1.py
1 import test as t
2
3 def main() :
4     t.jawne()
5     t.tajne()
6
7 if __name__ == "__main__" :
8     main()
```

```
main2.py
1 from test import *
2
3 def main() :
4     jawne()
5     tajne()
6
7 if __name__ == "__main__" :
8     main()
```

Maciek Gębala

Modularyzacja, Tablice dynamiczne

Notatki

Czytanie parametrów wywołania programów

We wszystkich językach programowania istnieją metody/biblioteki umożliwiające czytanie parametrów wywołania programów z linii poleceń (jako ciąg napisów).

Maciek Gębala

Modularyzacja, Tablice dynamiczne

Notatki

Implementacja w C

```
params.c
1 #include <stdio.h>
2
3 int main(int argc, char *argv[]) {
4     printf("%d\n", argc);
5
6     for (int i = 0; i < argc; i++)
7         printf("%s\n", argv[i]);
8
9     return 0;
10 }
```

W języku C nie ma typu napisowego - zastępuje go tablica elementów typu char. Do operowania na napisach konieczna jest biblioteka string.h zawierające m.in. takie funkcje jak strlen czy strcmp.

Maciek Gębala

Modularyzacja, Tablice dynamiczne

Notatki

Implementacja w Adzie

```

params.adb
with Ada.Text_IO; use Ada.Text_IO;
with Ada.Command_Line; use Ada.Command_Line;

procedure Params is
begin
  Put_Line (Command_Name);
  Put_Line (Argument_Count'Image);
  for i in 1 .. Argument_Count loop
    Put_Line (Argument (i)); -- funkcja, nie tablica
  end loop;
end Params;

```

Maciek Gębala Modularyzacja. Tablice dynamiczne

Implementacja w Pythonie

```
params.py
import sys

def main() :
    print(len(sys.argv))
    for i in range(len(sys.argv)) :
        print(sys.argv[i]);

if __name__ == "__main__":
    main()
```

Maciek Gębala Modularyzacja. Tablice dynamiczne

Tablica dynamiczna w C

- Do stworzenia tablicy dynamicznej w C potrzeba
 - zadeklarowania wskaźnika na typ elementów tablicy (unarny operator *);
 - użycia funkcji `malloc` do zarezerwowania odpowiedniego miejsca w pamięci;
 - użycia funkcji `free` do zwolnienia pamięci, gdy już nie potrzebujemy tablicy.

- zadeklarowania wskaźnika na typ elementów tablicy (unarny operator *);
- użycia funkcji `malloc` do zarezerwowania odpowiedniego miejsca w pamięci;
- użycia funkcji `free` do zwolnienia pamięci, gdy już nie potrzebujemy tablicy.

Maciek Gębala Modularyzacja. Tablice dynamiczne

Implementacja w C

```
primenumbers.h
#pragma once
#include <stdbool.h>

void compute_sieve(bool s[], unsigned n);

unsigned count_primes(bool s[], unsigned n);
```

Maciek Gębala Modularyzacja. Tablice dynamiczne

Notatki

Notatki

Notatki

Notatki

Implementacja w C

```
primenumbers.c
1 #include "primenumbers.h"
2
3 void compute_sieve(bool s[], unsigned n) {
4     unsigned i, j;
5     for ( i=2; i <= n; i++ ) s[i] = true;
6     for ( i=2; i <= n; i++ )
7         if ( s[i] )
8             for ( j=i+i; j <= n; j+=i )
9                 s[j] = false;
10 }
11
12 unsigned count_primes(bool s[], unsigned n) {
13     int i, c = 0;
14     for ( i=2; i <= n; i++ )
15         if ( s[i] ) c++;
16     return c;
17 }
```

Maciek Gębala

Modularyzacja, Tablice dynamiczne

Notatki

Implementacja w C

```
primenumberstest.c
1 #include <stdio.h>
2 #include <stdbool.h>
3 #include <stdlib.h>
4
5 #include "primenumbers.h"
6
7 int main(int argc, char *argv[]) {
8     unsigned long n, c;
9     bool *s;
10
11     if (argc != 2) {
12         printf("Zła liczba argumentów\n");
13         return -1;
14     }
15     sscanf(argv[1], "%lu", &n);
16     s = malloc((n+1)*sizeof(bool));
17     compute_sieve(s, n);
18     c = count_primes(s, n);
19     free(s);
20     printf("%lu\n", c);
21     return 0;
22 }
```

Maciek Gębala

Modularyzacja, Tablice dynamiczne

Notatki

Tablica dynamiczna w Adzie

Do stworzenia tablicy dynamicznej w Adzie potrzeba

- zadeklarowania typu wskaźnika na typ tablicy (słowo kluczowe `access`);
- zadeklarowania wskaźnika na typ tablicy;
- użycia operatora `new` do zarezerwowania odpowiedniego miejsca w pamięci;
- deklaracji i użycia funkcji `free` do zwolnienia pamięci, gdy już nie potrzebujemy tablicy.

Maciek Gębala

Modularyzacja, Tablice dynamiczne

Notatki

Implementacja w Adzie

```
primenumbers.ads
1 package PrimeNumbers is
2     type Sieve is array (Positive range <>) of Boolean;
3     type Sieve_Ptr is access Sieve;
4
5     procedure ComputeSieve (s : Sieve_Ptr);
6
7     function CountPrimes (s : Sieve_Ptr) return Natural;
8 end PrimeNumbers;
```

Maciek Gębala

Modularyzacja, Tablice dynamiczne

Notatki

Implementacja w Adzie

```
primenumbers.adb
1 package body PrimeNumbers is
2   procedure ComputeSieve (s : Sieve_Ptr) is
3     j : Natural;
4     begin
5       s.all := (others => True);
6       for i in s'Range loop
7         if s(i) then
8           j := i + i;
9           while j <= s'Last loop
10            s(j) := False; j := j + i;
11          end loop;
12        end if;
13      end loop;
14    end ComputeSieve;
15    function CountPrimes (s : Sieve_Ptr) return Natural is
16      c : Natural := 0;
17      begin
18        for i in s'Range loop
19          if s(i) then
20            c := c + 1;
21          end if;
22        end loop;
23        return c;
24      end CountPrimes;
25 end PrimeNumbers;
```

Maciek Gbala

Modularyzacja, Tablice dynamiczne

Notatki

Implementacja w Adzie

```
primenumberstest.adb
1 with Ada.Text_IO; use Ada.Text_IO;
2 with Ada.Command_Line; use Ada.Command_Line;
3 with Ada.Unchecked_Deallocation;
4 with PrimeNumbers; use PrimeNumbers;
5
6 procedure PrimeNumbersTest is
7   procedure Free is
8     new Standard.Ada.Unchecked_Deallocation (Sieve, Sieve_Ptr);
9   n : Natural;
10  c : Natural := 0;
11  s : Sieve_Ptr;
12  begin
13    if Argument_Count /= 1 then
14      Put_Line ("Zła liczba argumentów");
15      return;
16    end if;
17    n := Natural'Value (Argument (1));
18    s := new Sieve (2 .. n);
19    ComputeSieve (s);
20    c := CountPrimes (s);
21    Free (s);
22    Put_Line (c'Image);
23  end PrimeNumbersTest;
```

Maciek Gbala

Modularyzacja, Tablice dynamiczne

Notatki

Implementacja w Pythonie

W Pythonie nie ma tablic a lista jest typem dynamicznym.

```
packages/primenumbers.py
1 def create_sieve(s, n) :
2     for _ in range(n+1) :
3         s.append(True)
4
5 def compute_sieve(s) :
6     for i in range(2, len(s)) :
7         if s[i] :
8             j = i + i
9             while j < len(s) :
10              s[j] = False
11              j = j + i
12
13 def count_primes(s) :
14     c = 0
15     for i in range(2, len(s)) :
16         if s[i] :
17             c = c + 1
18     return c
```

Maciek Gbala

Modularyzacja, Tablice dynamiczne

Notatki

Implementacja w Pythonie

```
main.py
1 import sys
2
3 import packages.primenumbers
4
5 def main() :
6     if len(sys.argv) != 2 :
7         print("Zła liczba argumentów")
8         return
9
10    n = int(sys.argv[1])
11    s = []
12
13    packages.primenumbers.create_sieve(s, n)
14    packages.primenumbers.compute_sieve(s)
15    c = packages.primenumbers.count_primes(s)
16    print(c)
17
18 if __name__ == "__main__":
19     main()
```

Maciek Gbala

Modularyzacja, Tablice dynamiczne

Notatki

Implementacja w Pythonie

```
main1.py
1 import sys
2
3 import packages.primenumbers as pp
4
5 def main() :
6     if len(sys.argv) != 2 :
7         print("Zła liczba argumentów")
8         return
9
10    n = int(sys.argv[1])
11    s = []
12
13    pp.create_sieve(s, n)
14    pp.compute_sieve(s)
15    c = pp.count_primes(s)
16    print(c)
17
18 if __name__ == "__main__":
19     main()
```

Maciek Gębala

Modularyzacja, Tablice dynamiczne

Notatki

Implementacja w Pythonie

```
main2.py
1 from sys import argv
2
3 from packages.primenumbers import *
4
5 def main() :
6     if len(argv) != 2 :
7         print("Zła liczba argumentów")
8         return
9
10    n = int(argv[1])
11    s = []
12
13    create_sieve(s, n)
14    compute_sieve(s)
15    c = count_primes(s)
16    print(c)
17
18 if __name__ == "__main__":
19     main()
```

Maciek Gębala

Modularyzacja, Tablice dynamiczne

Notatki

Notatki

Notatki