

Techniki programowania

INP001002WI

rok akademicki 2018/19

semestr letni

Wykład 2

Karol Tarnowski

karol.tarnowski@pwr.edu.pl

A-1 p. 411B



Plan prezentacji (1)

- Systemy kontroli wersji
- Typy wyliczeniowe
- Tablice wielowymiarowe

Na podstawie:

- A. Allain, *Przewodnik dla początkujących C++*
- S. Prata, *Szkola programowania C++*



Plan prezentacji (2)

- Wskaźniki
- Referencje
- Dynamiczna alokacja pamięci
 - zmienne
 - tablice
 - tablice wielowymiarowe
- Możliwości typu **string**

Na podstawie:

- A. Allain, *Przewodnik dla początkujących C++*
- S. Prata, *Szkoła programowania C++*



Systemy kontroli wersji


- W pracy nad projektami informatycznymi stosuje się systemy kontroli wersji
- Systemy kontroli wersji służą do śledzenia zmian głównie w kodzie źródłowym oraz wspierają proces łączenia zmian dokonanych przez wiele osób



Systemy kontroli wersji

- Programy przygotowane w ramach kursu należy przechowywać w postaci repozytorium kodu
- Istnieją różne systemy kontroli wersji i narzędzie do obsługi repozytoriów
- Do przechowywania programów należy wykorzystać serwis Bitbucket (bitbucket.org)

Bitbucket

 Meet Bitbucket Pipes. 35+ new ways to automate your CI/CD pipeline. [Learn more](#)

Built for professional teams

Bitbucket is more than just Git code management. Bitbucket gives teams one place to plan projects, collaborate on code, test, and deploy.

[Get started for free](#)

Or host it yourself with [Bitbucket Enterprise](#) →

bugfix/123-bug remove extra padding

Approve

Merge



bugfix/123-bug → master [OPEN](#)

I made a few changes to tidy up the code. Please let me know if all looks good!



js / core.js

Bitbucket

[Features](#)[Integrations](#)[Server](#)[Data Center](#)[Pricing](#)[Log in](#)[Get started](#)

Create your account

Enter your email address

Continue

[Blog](#) · [Support](#) · [Plans & pricing](#) · [Documentation](#) · [API](#) · [Site status](#) · [Cloud terms of service](#) · [Privacy policy](#)

[Jira Software](#) · [Confluence](#) · [Bamboo](#) · [Sourcetree](#)

 **ATLASSIAN**

Bitbucket



[Features](#)

[Integrations](#)

[Server](#)

[Data Center](#)

[Pricing](#)

[Log in](#)

[Get started](#)

Check your inbox to verify your email

We've sent an email to [\[redacted\]](#)

Follow the instructions to verify your email address.



Didn't receive the verification email? [Send it again](#)

Email verification helps us to ensure your data will always be safe



Politechnika
Wroclawska

Bitbucket



[Features](#)

[Integrations](#)

[Server](#)

[Data Center](#)

[Pricing](#)

[Log in](#)

[Get started](#)

Almost done

Create a unique username for Bitbucket Cloud

bitbucket.org /

[Continue](#)

You're signing up with your Atlassian account for [k...](#)

Not the account you want to use? [Log out](#) and enter the correct email address.

[Blog](#) · [Support](#) · [Plans & pricing](#) · [Documentation](#) · [API](#) · [Site status](#) · [Cloud terms of service](#) · [Privacy policy](#)

[Jira Software](#) · [Confluence](#) · [Bamboo](#) · [Sourcetree](#)

Bitbucket



Before we drop you in Bitbucket, help us tailor your experience

What describes your experience with source control? I have no experience

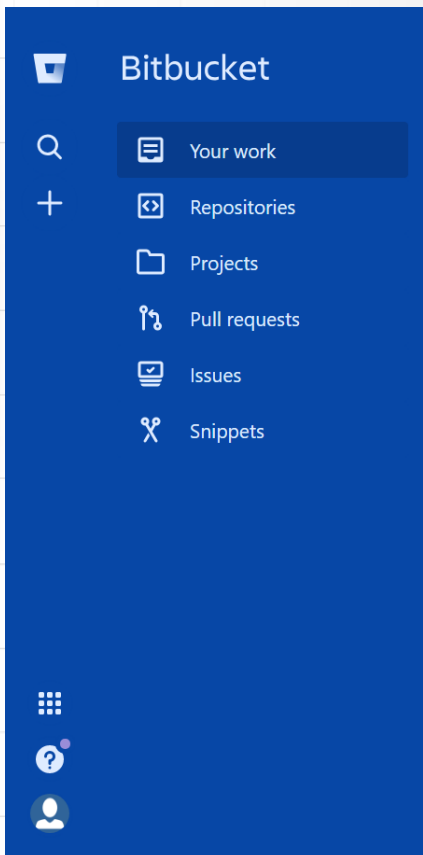
How many people do you think you'll work with on Bitbucket? 1-5

What best describes your role? Student

Skip

Submit

Bitbucket



Bitbucket

- 🔍 Your work
- + 📁 Repositories
- 📁 Projects
- 🔗 Pull requests
- 📄 Issues
- ✂ Snippets

☰ ? 👤

Your work



Here's where your work shines through

Set up a repository to get going with your code. After that, you'll find your relevant repositories and work right here.

Create repository

Import repository



Bitbucket



Create a new repository

[Import repository](#)

Repository name ^{*}

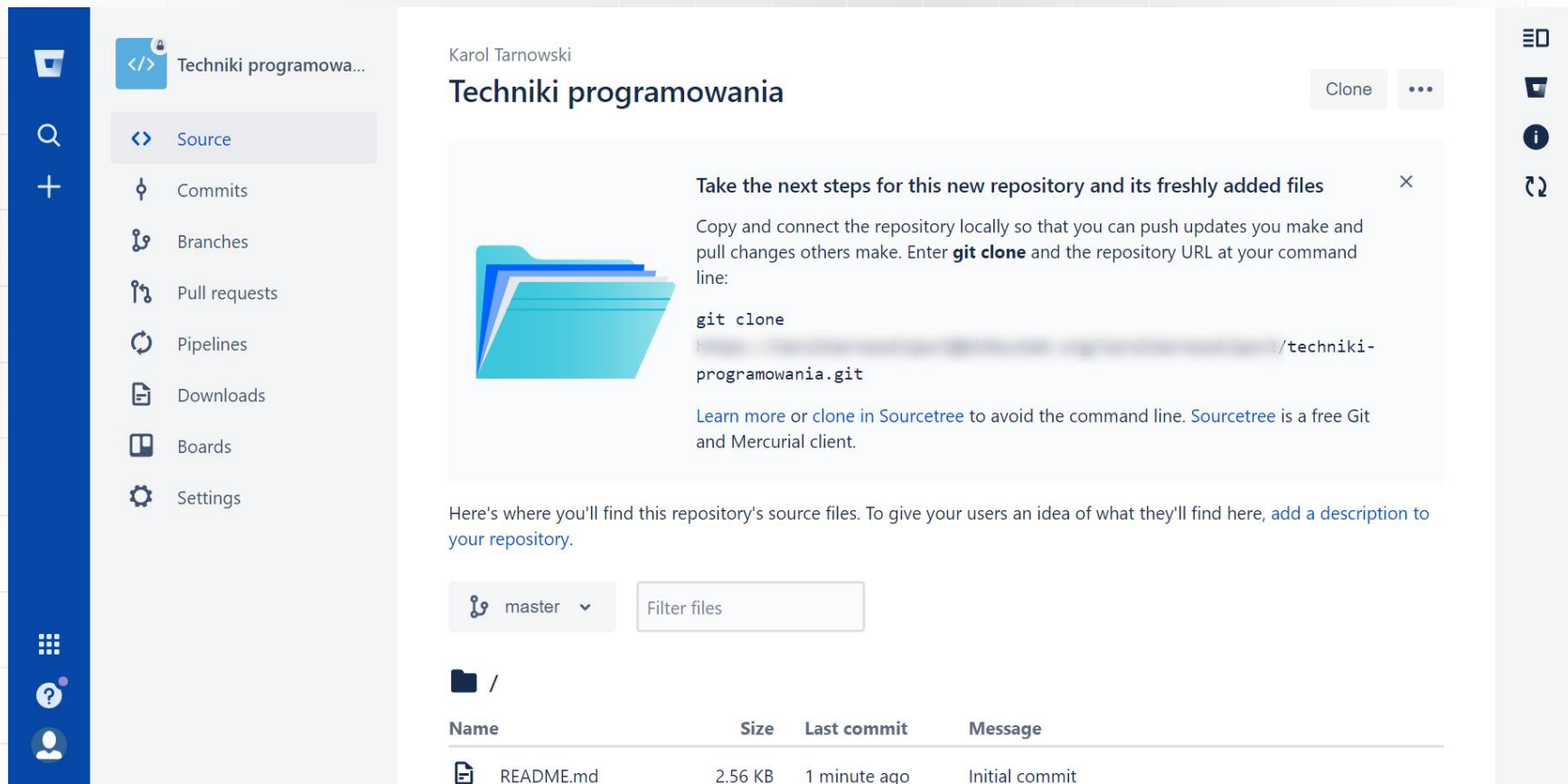
Access level This is a private repository
Uncheck to make this repository public. Public repositories are open source and can be viewed by anyone.

Include a README? ▼

Version control system Git
 Mercurial

[> Advanced settings](#)

Bitbucket



The screenshot shows the Bitbucket interface for a new repository named "Techniki programowania" by Karol Tarnowski. The left sidebar contains navigation options: Source (selected), Commits, Branches, Pull requests, Pipelines, Downloads, Boards, and Settings. The main content area features a "Clone" button and a "..." menu. A prominent instruction box guides the user on how to clone the repository locally using the `git clone` command, providing a placeholder for the repository URL. Below this, there is a section for adding a repository description. At the bottom, a file browser shows the "master" branch with a "Filter files" input field. A table lists the repository's files, currently showing only "README.md" with a size of 2.56 KB, committed 1 minute ago.

Techniki programowania

Clone ...

Take the next steps for this new repository and its freshly added files

Copy and connect the repository locally so that you can push updates you make and pull changes others make. Enter **git clone** and the repository URL at your command line:

```
git clone [redacted] /techniki-programowania.git
```

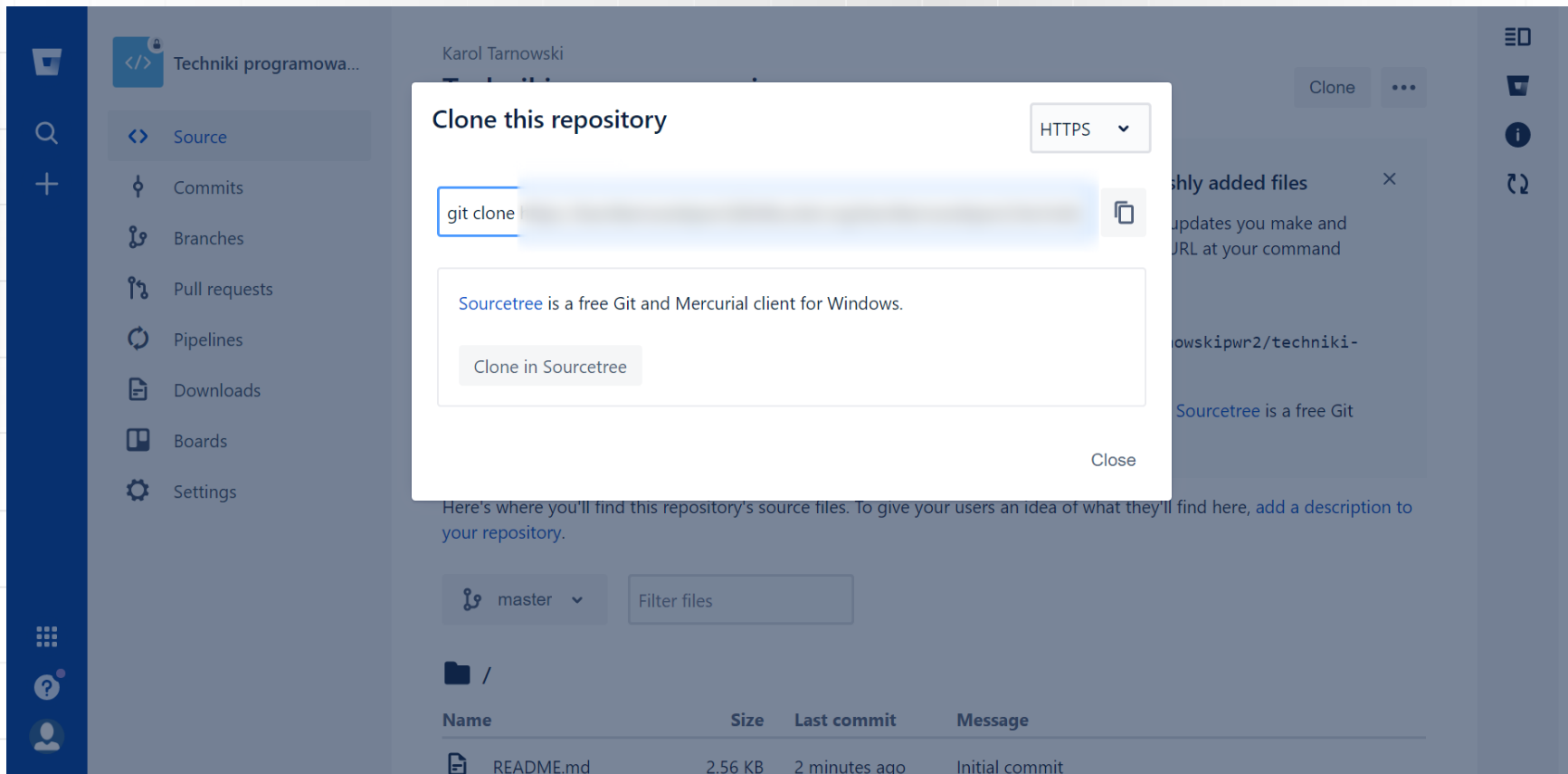
Learn more or clone in Sourcetree to avoid the command line. Sourcetree is a free Git and Mercurial client.

Here's where you'll find this repository's source files. To give your users an idea of what they'll find here, [add a description to your repository.](#)

master Filter files

Name	Size	Last commit	Message
README.md	2.56 KB	1 minute ago	Initial commit

Bitbucket



The screenshot shows the Bitbucket web interface. A modal dialog box titled "Clone this repository" is centered on the screen. The dialog has a "Clone" button in the top right corner. Below the title, there is a dropdown menu set to "HTTPS". A text input field contains the command "git clone" followed by a blurred URL. To the right of the input field is a copy icon. Below the input field, there is a text box containing the text: "Sourcetree is a free Git and Mercurial client for Windows." Below this text box is a button labeled "Clone in Sourcetree". At the bottom right of the dialog is a "Close" button. The background interface shows a sidebar with navigation options: Source, Commits, Branches, Pull requests, Pipelines, Downloads, Boards, and Settings. The main content area shows the repository name "Karol Tarnowski" and a list of files with columns for Name, Size, Last commit, and Message. The file "README.md" is listed with a size of 2.56 KB and a last commit of 2 minutes ago.

Clone this repository

HTTPS

git clone

Sourcetree is a free Git and Mercurial client for Windows.

Clone in Sourcetree

Close

Here's where you'll find this repository's source files. To give your users an idea of what they'll find here, add a description to your repository.

master

Filter files

Name	Size	Last commit	Message
README.md	2.56 KB	2 minutes ago	Initial commit



Sourcetree

The screenshot shows the Sourcetree application window with a blue title bar and menu. The main window displays the 'Clone' dialog box. The dialog has a title 'Clone' and a subtitle 'Cloning is even easier if you set up a [remote account](#)'. It contains three input fields: a repository URL field (empty), a local folder path field containing 'C:\Users\karol\Documents\techniki-programowania', and a local folder name field containing 'techniki-programowania'. There are 'Browse' buttons next to the first two fields. Below the fields is a 'Local Folder:' label and a dropdown menu showing '[Root]'. At the bottom, there is a 'Advanced Options' section with a downward arrow and a large blue 'Clone' button.

File Edit View Repository Actions Tools Help

techniki-programowania New tab

Local Remote Clone Add Create

Clone

Cloning is even easier if you set up a [remote account](#)

Repository Type: This is a Git repository

C:\Users\karol\Documents\techniki-programowania Browse

techniki-programowania

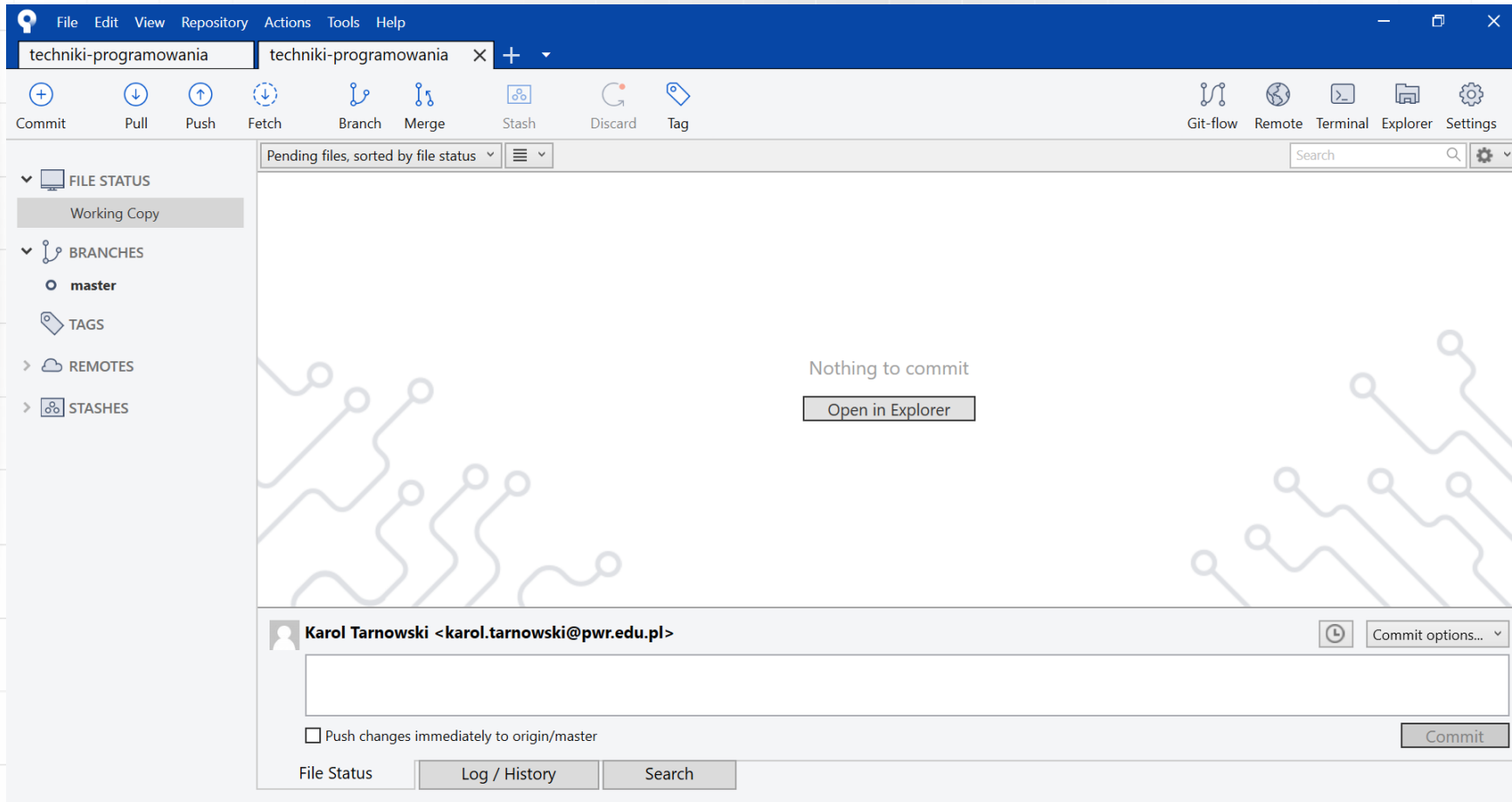
Local Folder:

[Root]

Advanced Options

Clone

Sourcetree





Typy wyliczeniowe

- Typ wyliczeniowy definiowany jest słowem kluczowym `enum`
- Wszystkie możliwe wartości znajdują się na liście
`enum nazwa_typu{lista_wartosci};`

- Przykład

```
enum card_suit{  
    SPADE, HEART, DIAMOND, CLUB  
};
```



Typy wyliczeniowe

```
Start here x enum_example.cpp x
1  #include <iostream>
2  #include <cstdlib> //wykorzystywane funkcje: srand(), rand()
3  #include <ctime>   //wykorzystywane funkcje: time()
4
5  using namespace std;
6
7  //deklaracja typu wyliczeniowego możliwych kolorów kart
8  enum card_suit{
9      SPADE, HEART, DIAMOND, CLUB
10 };
11
12 //deklaracja typu wyliczeniowego możliwych wartości kart
13 enum card_rank{
14     TWO, THREE, FOUR, FIVE, SIX, SEVEN, EIGHT, NINE, TEN, JACK, QUEEN, KING, ACE
15 };
16
17 /*stałe tablice globalne łańcuchów znakowych wykorzystywane
18 przy wypisywaniu wartości kart*/
19 const char* card_suit_names[] = \
20     {"pik ", "kier ", "karo ", "trefl" };
21 const char* card_rank_names[] = \
22     {" 2", " 3", " 4", " 5", " 6", " 7", " 8", " 9", "10", " J", " Q", " K", " A" };
23
```



Typy wyliczeniowe

```
Start here x enum_example.cpp x
24 int main() {
25     //inicjalizacja generatora liczb pseudolosowych
26     srand(time(NULL));
27
28     //losowa wartość całkowita z przedziału 0-3 (typ int)
29     //jest rzutowana na typ card_suit
30     card_suit c_suit = card_suit(rand()%4);
31     //przypisanie c_suit = rand()%4 nie byłoby prawidłowe
32
33     //wybór losowej wartości karty
34     card_rank c_rank = card_rank(rand()%13);
35
36     //wyświetlenie wartości karty z wykorzystaniem nazw kolorów i wartości
37     cout << card_rank_names[c_rank] << " " << card_suit_names[c_suit] << endl;
38
39     //wyświetlenie wartości kart
40     cout << c_rank << " " << c_suit << endl;
41
42     return 0;
43 }
```



Typy wyliczeniowe

- Zaletą typów wyliczeniowych może być zwiększenie czytelności kodu

```
card_suit suit = HEART;
```

```
...
```

```
switch(suit) {
```

```
    case SPADE: //wybrano pik
```

```
    case HEART: //wybrano kier
```

```
    case DIAMOND: //wybrano karo
```

```
    case CLUB: //wybrano trefl
```

```
}
```



Tablice wielowymiarowe

- Deklaracja tablicy wielowymiarowej

```
double matrix_A[3][4];  
double matrix_B[3][4] = \  
    { {0, 0, 1, -2}, \  
      {0, 1, -2, 1}, \  
      {1, -2, 1, 0} };
```



Tablice wielowymiarowe

- Dostęp do elementów tablicy

```
for(int i=0; i<3; i++){  
    for(int j=0; j<4; j++){  
        cout << matrix_A[i][j] << " " ;  
    }  
    cout << endl;  
}
```



Tablice wielowymiarowe

- Przekazanie tablicy wielowymiarowej do funkcji

```
void add_matrix(double a[][4], double b[][4]) {  
    for(int i=0; i<3; i++)  
        for(int j=0; j<4; j++)  
            a[i][j] += b[i][j];  
}
```



Tablice wielowymiarowe

```
Start here x multidimensional_array_example.cpp x
1  /*multidimensional_array_example.cpp*/
2  /*Program pokazujący możliwości działania
3  na tablicach wielowymiarowych.*/
4
5  #include <iostream>
6
7  using namespace std;
8
9  /*Deklaracje funkcji pobierających jako argument tablice wielowymiarowe.
10 Należy podać rozmiary tablicy - wszystkie poza pierwszym.
11 Pierwszy rozmiar tablicy można podać.*/
12 void print_matrix(double [3][4]);
13 void add_matrix(double [][][4], double [][][4]);
14
15 int main(){
16     //Deklaracje tablic rozmiaru 3x4, połączone z inicjalizacją.
17     double matrix_A[3][4] = { {-2, 1, 0, 0}, {1, -2, 1, 0}, {0, 1, -2, 1} };
18     double matrix_B[3][4] = { {0, 0, 1, -2}, {0, 1, -2, 1}, {1, -2, 1, 0} };
19
```




Tablice wielowymiarowe

```
Start here x multidimensional_array_example.cpp x
20 //Wyświetlenie zawartości tablicy
21 cout << "macierz A" << endl;
22 for(int i=0; i<3; i++){
23     for(int j=0; j<4; j++)
24         /*Odwołanie do elementu tablicy następuje przez podanie
25         indeksów we wszystkich kierunkach.*/
26         cout << matrix_A[i][j] << " ";
27     cout << endl;
28 }
29
30 //Wyświetlenie zawartości tablicy z wykorzystaniem funkcji.
31 cout << "macierz B" << endl;
32 print_matrix(matrix_B);
33
34 //Wywołanie funkcji obliczającej sumę dwóch macierzy.
35 add_matrix(matrix_A, matrix_B);
36
37 cout << "macierz A" << endl;
38 print_matrix(matrix_A);
39 }
40
```

Tablice wielowymiarowe

```
Start here x multidimensional_array_example.cpp x
41 void print_matrix(double matrix[3][4]){
42     for(int i=0; i<3; i++){
43         for(int j=0; j<4; j++)
44             cout << matrix[i][j] << " ";
45         cout << endl;
46     }
47 }
48
49 /*Funkcja dodająca macierz b do macierzy a.
50 Elementy macierzy a są aktualizowane
51 (tablice przekazane przez referencję)*/
52 void add_matrix(double a[][4], double b[][4]){
53     for(int i=0; i<3; i++)
54         for(int j=0; j<4; j++)
55             a[i][j] += b[i][j];
56 }
```



Tablice wielowymiarowe

```
double a[3][2];
```

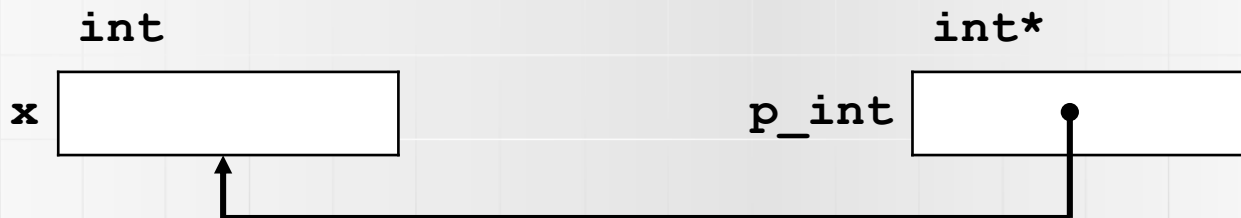
a[0][0]	a[0][1]
a[1][0]	a[1][1]
a[2][0]	a[2][1]

a[0][0]	a[0][1]	a[1][0]	a[1][1]	a[2][0]	a[2][1]
---------	---------	---------	---------	---------	---------



Wskaźniki - przypomnienie

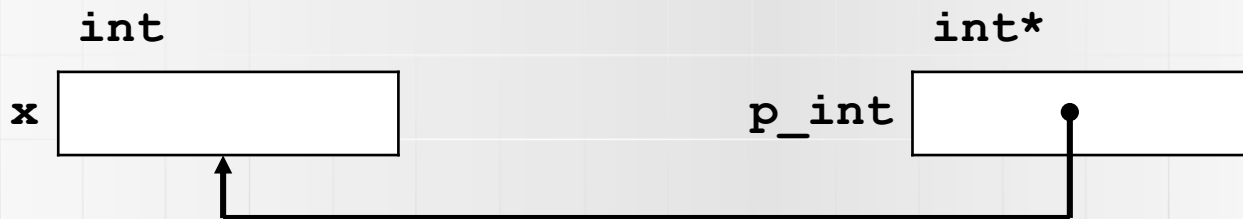
```
Start here x pointer.cpp x
1  /*pointer.cpp*/
2  /*Program pokazujący wykorzystanie operatorów
3  adresu i wskaźnikowego.*/
4
5  #include<iostream>
6
7  using namespace std;
8
9  int main(){
10     int x;           //deklaracja zmiennej całkowitej
11     int* p_int;     //deklaracja zmiennej wskaźnikowej
12     //obie zmienne nie zostały zainicjalizowane
13
14     p_int = &x;     /*operatorem adresu odczytujemy, gdzie w pamięci
15     znajduje się zmienna x, a następnie zapisujemy ten adres w zmiennej p_int*/
16
```





Wskaźniki - przypomnienie

```
Start here × pointer.cpp ×
17     cout << "Podaj liczbe: "; //wczytanie wartosci do zmiennej x
18     cin >> x;
19
20     cout << *p_int << endl;    /*wyświetlenie zmiennej x z wykorzystaniem
21     operatora wskaźnikowego, który pozwala nam odczytać wartość spod adresu;
22     w tym przypadku to zmienna x, bo p_int pokazuje na x (linia 14)*/
23
24     *p_int = 10;    /*przypisanie z wykorzystaniem operatora wskaźnikowego,
25     modyfikowana jest zmienna x, bo p_int pokazuje na x.*/
26
27     cout << x;    //wyświetlana jest liczba 10
28 }
```





Referencje

- O referencji można myśleć jako o stałym wskaźniku, dla którego nie potrzebujemy jawnie posługiwać się operatorem wskaźnikowym.
- Jeżeli referencja ma być stała, to musi być inicjalizowana w chwili deklaracji.
- Podstawowym zastosowaniem referencji jest przekazywanie dużych struktur danych do funkcji.

```
int x = 5;
```

```
int &ref = x;
```



Wskaźniki, referencje i funkcje

```
Start here x swap.cpp x
1  /*swap.cpp*/
2  /*Program pokazujący różne sposoby przekazywania
3  argumentów do funkcji:
4  - przez wartość (kopie),
5  - przez wskaźnik,
6  - przez referencję.*/*
7
8  #include <iostream>
9
10 using namespace std;
11
12 void swap_value(int left, int right);
13 void swap_pointer(int* left, int* right);
14 void swap_reference(int& left, int& right);
15
16 int main(){
17     int x = 1, y = 2;
18     swap_value(x,y);
19     cout << x << " " << y << endl;
20     swap_pointer(&x,&y);
21     cout << x << " " << y << endl;
22     swap_reference(x,y);
23     cout << x << " " << y << endl;
24 }
```



Wskaźniki, referencje i funkcje

```
Start here x swap.cpp x
26  /*Funkcja pobiera kopie dwóch liczb całkowitych.
27     Oryginały się nie zamieniają, gdyż funkcja
28     operuje na kopiach.*/
29  void swap_value(int left, int right){
30     int temp = left;
31     left = right;
32     right = temp;
33  }
34
35  /*Funkcja pobiera wskaźniki do dwóch liczb całkowitych.
36     Wykorzystanie operatora wskaźnikowego pozwala
37     zamienić miejscami oryginały.*/
38  void swap_pointer(int* left, int* right){
39     int temp = *left;
40     *left = *right;
41     *right = temp;
42  }
43
44  /*Funkcja pobiera referencje do dwóch liczb całkowitych.
45     Nie potrzebujemy operatora wskaźnikowego,
46     ale działamy na oryginałach argumentów.*/
47  void swap_reference(int& left, int& right){
48     int temp = left;
49     left = right;
50     right = temp;
51  }
```




Dynamiczna alokacja pamięci

- Alokacja pojedynczej zmiennej

```
typ* pointer = new typ;
```

```
int* p_int = new int;
```

- Zwalnianie pamięci

```
delete p_int;
```

- Dobrą praktyką jest zerowanie nieużywanego wskaźnika

```
delete p_int;
```

```
p_int = NULL;
```



Dynamiczna alokacja pamięci

- Alokacja tablicy

```
typ* pointer = new typ[rozmiar];
```

```
int* p_int = new int[16];
```

- Zwalnianie tablicy

```
delete[] p_int;
```



Dynamiczna alokacja pamięci

```
Start here × resize_table.cpp ×
1  /*resize_table.cpp*/
2   /*Przykładowy program wykorzystujący
3     dynamiczną alokację pamięci.
4     W dynamicznie alokowanej tablicy numbers
5     zbierane są liczby podane przez użytkownika.
6     Tablica ma długość length,
7     a zmienna next przechowuje liczbę
8     wpisanych danych (jednocześnie
9     indeks następnego elementu).*/
10
11  #include <iostream>
12
13  using namespace std;
14
15   /*Funkcja drukuje informacje o tablicy numbers:
16     jej długość (length), liczbę elementów (filled),
17     oraz zawartość.*/
18  void printTable(int* numbers, int length, int filled);
19
20   /*Funkcja przepisuje tablicę numbers do
21     nowoalokowanej tablicy o dwa razy większej długości.
22     Zwalnia nie używaną pamięć i zwraca adres
23     nowej tablicy, oraz aktualizuje długość (length)
24     przekazaną przez referencję.*/
25  int* enlargeTable(int* numbers, int& length);
```



Dynamiczna alokacja pamięci

```
Start here × resize_table.cpp ×
27 int main(){
28     int next = 0;    //początkowo nie ma elementów w tablicy
29     int length = 8; //początkowa długość = 8
30     int *numbers = new int[length]; //odpowiednia alokacja
31     int num;
32     cout << "Podaj liczbe (0 - wyjscie z programu): ";
33     cin  >> num;
34     while(num!=0){
35         //jeśli kolejny element nie zmieści się w tablicy
36         if( length == next ){
37             //to powiększ tablicę
38             numbers = enlargeTable(numbers, length);
39         }
40         //wpisz nowy element do tablicy i przesun indeks next
41         numbers[next++] = num;
42         cout << "Aktualny stan tablicy" << endl;
43         //wyświetlenie informacji o tablicy
44         printTable(numbers,length,next);
45         cout << "Podaj liczbe (0 - wyjscie z programu): ";
46         cin  >> num;
47     }
48     //zwolnienie pamięci
49     delete[] numbers;
50 }
```



Dynamiczna alokacja pamięci

Start here × **resize_table.cpp** ×

```
61  int* enlargeTable(int* numbers, int& length){
62      //alokacja nowej - dwa razy większej tablicy
63      int* newNumbers = new int[length*2];
64      //przepisanie elementów do nowej tablicy
65      for(int i=0; i<length; i++){
66          newNumbers[i] = numbers[i];
67      }
68      //aktualizacja długości
69      length *= 2;
70      //zwolnienie starej pamięci
71      delete[] numbers;
72      //zwracana jest nowa tablica
73      return newNumbers;
74 }
```



Dynamiczna alokacja pamięci

```
/*pascals_triangle.cpp*/
```

```
int **triangle;  
...  
triangle = new int* [n];  
...  
for(int i=0; i<n; i++){  
    triangle[i] = new int[i+1];  
    ...  
}  
...  
for(int i=0; i<n; i++){  
    delete[] triangle[i];  
}  
delete[] triangle;
```



Możliwości typu `string`

- metody `size()` oraz `length()` zwracają długość łańcucha
- dostęp do znaków poprzez operator `[]` lub metodę `at()`

```
string s = "moj lancuch"  
for(int i=0; i<s.length(); i++) {  
    cout << s[i];  
}
```



Możliwości typu `string`

- metoda `substr()` tworzy łańcuch będący kopią wskazanego fragmentu

```
string s = "moj lancuch";
```

```
string p = s.substr(2,3); // p = "j l"
```


Przekazywanie łańcuchów przez referencję

```
void drukujLancuch(const string& s) {  
    cout << s;  
    //s = "abc"; //niedozwolone  
}
```



Podsumowanie

- Systemy kontroli wersji
- Typy wyliczeniowe
- Tablice wielowymiarowe
- Wskaźniki
- Referencje
- Dynamiczna alokacja pamięci
- Możliwości typu `string`