



Kierunek: Grafika komputerowa i elementy rzeczywistości mieszanej / Informatyka

Rok akademicki: 2022/2023

Przedmiot: Podstawy programowania zorientowane obiektowo

Semestr: II

Grupa dziekańska: 1

Sprawozdanie z: Laboratorium numer 3

data: 10 maja 2023

Grupa robocza: GrupaRobocza06

Skład grupy:

Osoba 1: Rostyslav Hrenitskyi

Osoba 2: Maksym Szczurko

Osoba 3 : Wojciech Gawlas

Osoba 4:

Aktywności realizowane przez członków grupy roboczej, czyli kto i za co był odpowiedzialny podczas laboratorium/zadania domowego oraz utworzenia sprawozdania (min 3 na osobę):

Osoba 1: Rostyslav Hrenitskyi

Zadanie 1, 2, 2a, 3, 4, 5, 5a, 5b, 5c, 6, 7.

Osoba 2: Maksym Szczurko

Zadanie 1, 2, 2a, 3, 4, 5, 5a, 5b, 5c, 6, 7.

Osoba 3: Wojciech Gawlas

Zadanie 1, 2, 2a, 3, 4, 5, 5a, 5b, 5c, 6, 7.

Osoba 4:

Sposoby i narzędzia komunikacji grupy roboczej:

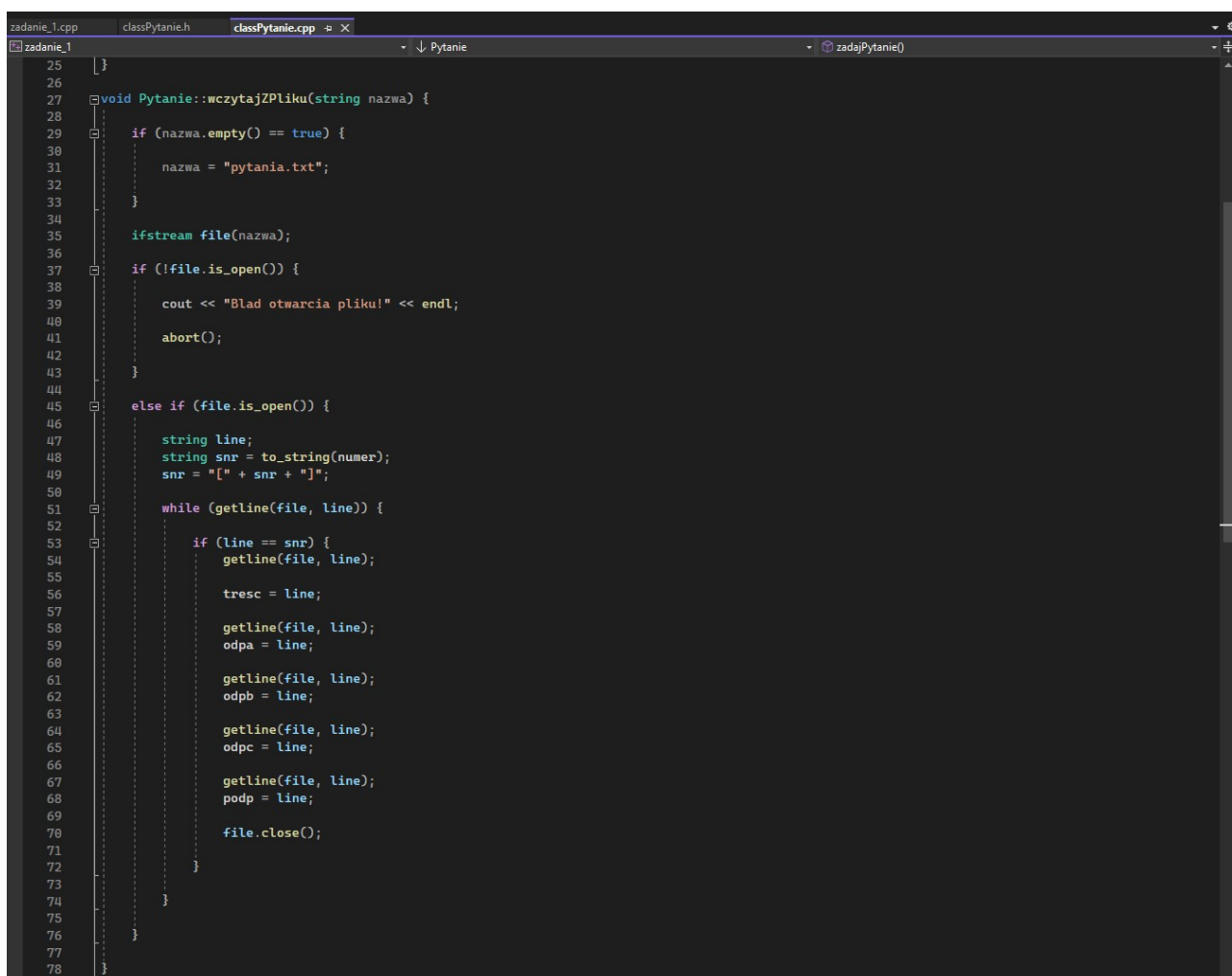
GitHub.

Zadanie 1

Patrząc na nasz program QUIZZ z modułu poprzedniego (ten podzielony już na pliki) należy dodać jeszcze możliwość podania nazwy pliku w trakcie pracy programu, ponieważ aktualnie jest ustalona w kodzie, na sztywno. Zastanowić się czy nie przydała by się możliwość automatycznego ustalania nazwy pliku podczas tworzenia obiektu. Taka możliwość nazywa się konstruktorem.

Rozwiązanie zadania 1

Patrząc na nasz program QUIZZ z modułu poprzedniego (ten podzielony już na pliki) należy dodać jeszcze możliwość podania nazwy pliku w trakcie pracy programu, ponieważ aktualnie jest ustalona w kodzie, na sztywno. Zastanowić się czy nie przydała by się możliwość automatycznego ustalania nazwy pliku podczas tworzenia obiektu.



```
25 }
26
27 void Pytanie::wczytajZPliku(string nazwa) {
28     if (nazwa.empty() == true) {
29         nazwa = "pytania.txt";
30     }
31     ifstream file(nazwa);
32     if (!file.is_open()) {
33         cout << "Bład otwarcia pliku!" << endl;
34         abort();
35     }
36     else if (file.is_open()) {
37         string line;
38         string snr = to_string(number);
39         snr = "[" + snr + "]";
40         while (getline(file, line)) {
41             if (line == snr) {
42                 getline(file, line);
43                 tresc = line;
44                 getline(file, line);
45                 odpa = line;
46                 getline(file, line);
47                 odpb = line;
48                 getline(file, line);
49                 odpc = line;
50                 getline(file, line);
51                 podp = line;
52                 file.close();
53             }
54         }
55     }
56 }
```

source: ...\\Modul 3\\Labaratorium 3\\Zadanie 1\\classPytanie.

```
zadanie_1.cpp  classPytanie.h  classPytanie.cpp
zadanie_1 (Global Scope)  main()
1  #include "classPytanie.h"
2  #include <iostream>
3
4  using namespace std;
5
6  int main() {
7
8      Pytanie P1, P2, P3;
9
10     P1.ustawNumer(1);
11     P1.wczytajZPliku("");
12     P1.zadajPytanie();
13     P1.badajOdpowiedz();
14
15     P2.ustawNumer(2);
16     P2.wczytajZPliku("");
17     P2.zadajPytanie();
18     P2.badajOdpowiedz();
19
20     P3.ustawNumer(3);
21     P3.wczytajZPliku("");
22     P3.zadajPytanie();
23     P3.badajOdpowiedz();
24
25
26     cout << "Punkty: " << P1.badajOdpowiedz() + P2.badajOdpowiedz() + P3.badajOdpowiedz() << " " << endl;
27
28     return (0);
29
30 }
```

source: ...\\Modul 3\\Labaratorium 3\\Zadanie 1\\zadanie_1.cpp

Wyjście(P1, P2, P3.wczytajZPliku("")):

```
Microsoft Visual Studio Debug
Pytanie numer 1:
Jaki odglos wydaje kot?

a. miau
b. hau
c. muu

Twoja odpowiedz: a

Odpowiedz zapisana.

Pytanie numer 2:
Kto jest najwazniejszy w polsce?

a. prezydent
b. prezes
c. premier

Twoja odpowiedz: b

Odpowiedz zapisana.

Pytanie numer 3:
Jezyk JAVA to jezyk poziomu

a. wysokiego
b. maszynowy
c. posredni

Twoja odpowiedz: c

Odpowiedz zapisana.

Punkty: 3.

C:\Users\qawith\Desktop\1\CPP\Semestr 2\PPK IBrzezek\Modul 3\Labaratorium 3\Zadanie 1\Debug\zadanie_1.exe (process 18276) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

source: ...\\Modul 3\\Labaratorium 3\\Zadanie 1\\classPytanie.cpp

Wyjście(P1, P2, P3.wczytajZPliku("pytania.txt")):

```
Microsoft Visual Studio Debu x + | v
Pytanie numer 1:
Jaki odglos wydaje kot?

a. miau
b. hau
c. muu

Twoja odpowiedz: a

Odpowiedz zapisana.

Pytanie numer 2:
Kto jest najwazniejszy w polsce?

a. prezydent
b. prezes
c. premier

Twoja odpowiedz: b

Odpowiedz zapisana.

Pytanie numer 3:
Jezyk JAVA to jezyk poziomu

a. wysokiego
b. maszynowy
c. posredni

Twoja odpowiedz: c

Odpowiedz zapisana.

Punkty: 3.

C:\Users\qawith\Desktop\1\CPP\Semestr 2\PPK IBrzezek\Modul 3\Labaratorium 3\Zadanie 1\x64\Debug\zadanie_1.exe (process 11132) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

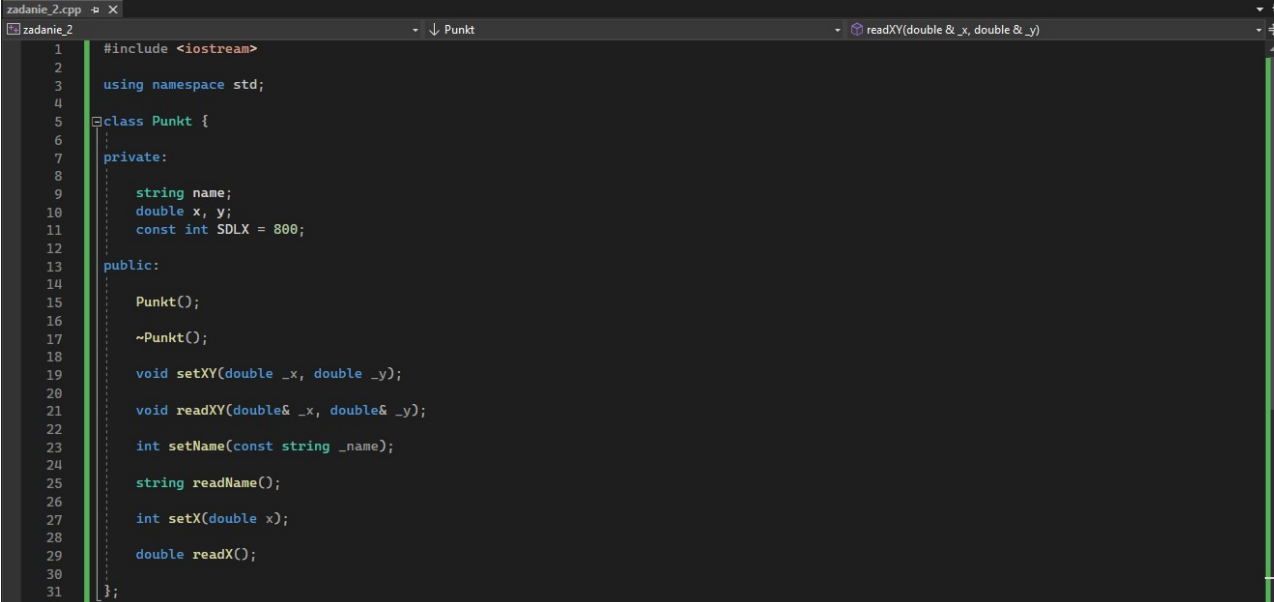
Zadanie 2

Stosując jeden plik kodu, napisać prosty program prezentujący idee konstruktora i destruktor. Klasa o nazwie Punkt przechowuje współrzędne punktu na płaszczyźnie oraz metodę je ustalającą oraz odczytującą. Pojawia się tu też konstruktor inicjalizujący współrzędne punktu podanymi, podczas tworzenia obiektu, wartościami współrzędnych. Kolejno argumenty konstruktora zasilić wartościami domyślnymi. Omówić tematykę konstruktorów przeciążonych i wartości domyślnych. Utworzyć dwa obiekty używając konstruktorów w różnych postaciach. Omówić słowo kluczowe this na przykładzie inicjalizacji cech obiektu argumentami konstruktora posiadającymi tę samą nazwę jak cechy. Zastanowić się jak tworzone i przechowywane są obiekty. Czy obiekty utworzone tak jak każda inna zmienna są w trakcie pracy kasowane gdy programista ich nie używa?

Rozwiązanie zadania 2

Stosując jeden plik kodu, napisać prosty program prezentujący idee konstruktora i destruktor. Klasa o nazwie Punkt przechowuje współrzędne punktu na płaszczyźnie oraz metodę je ustalającą oraz odczytującą. Pojawia się tu też konstruktor inicjalizujący współrzędne punktu podanymi, podczas tworzenia obiektu, wartościami współrzędnych. Kolejno argumenty konstruktora zasilić wartościami domyślnymi. Omówić tematykę konstruktorów przeciążonych i wartości domyślnych. Utworzyć dwa obiekty używając konstruktorów w różnych postaciach.

Klasa:



```
1  #include <iostream>
2
3  using namespace std;
4
5  class Punkt {
6
7  private:
8
9      string name;
10     double x, y;
11     const int SDLX = 800;
12
13  public:
14
15     Punkt();
16
17     ~Punkt();
18
19     void setXY(double _x, double _y);
20
21     void readXY(double& _x, double& _y);
22
23     int setName(const string _name);
24
25     string readName();
26
27     int setX(double x);
28
29     double readX();
30
31 };
32
```

source: ...\\Modul 3\\Laboratorium 3\\Zadanie 2\\zadanie_2.cpp

Main:

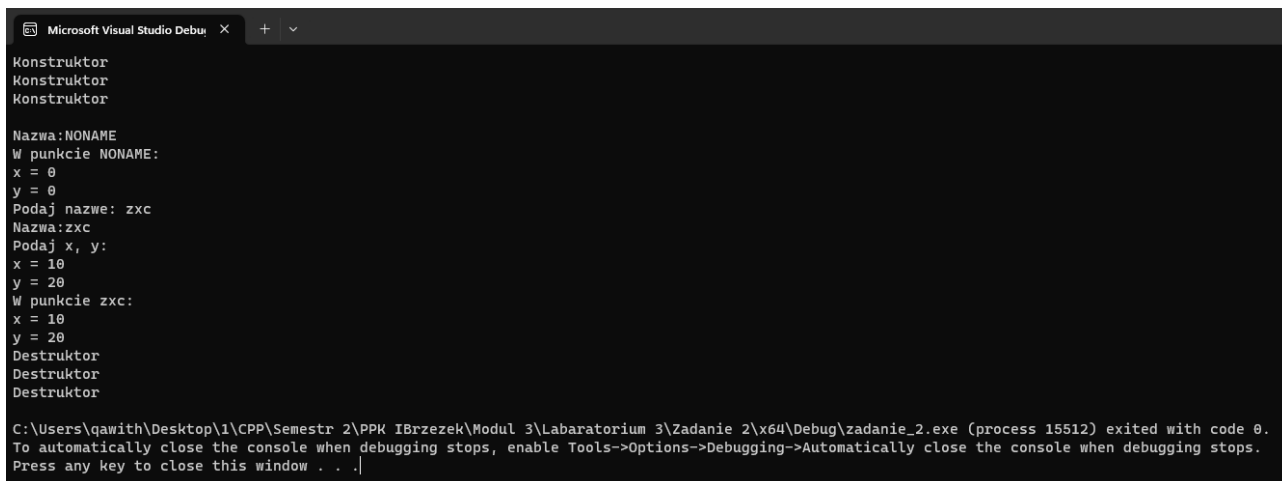
```
zadanie_2.cpp x
zadanie_2 (Global Scope) main()
31 };
32
33 int main() {
34
35     string nazwa;
36     double x, y;
37     Punkt A, B, C;
38
39     cout << endl << "Nazwa:" << B.readName() << endl;
40
41     B.readXY(x, y);
42     cout << "W punkcie " << B.readName() << ":" << endl;
43     cout << "x = " << x << endl;
44     cout << "y = " << y << endl;
45
46     cout << "Podaj nazwe: "; cin >> nazwa;
47     A.setName(nazwa);
48     if (A.setName(nazwa) == -1) {
49         cout << "Bledna nazwa!" << endl;
50         exit(10);
51     }
52     cout << "Nazwa:" << A.readName() << endl;
53
54     cout << "Podaj x, y: " << endl;
55     cout << "x = "; cin >> x;
56     cout << "y = "; cin >> y;
57     A.setXY(x, y);
58
59     A.readXY(x, y);
60     cout << "W punkcie " << A.readName() << ":" << endl;
61     cout << "x = " << x << endl;
62     cout << "y = " << y << endl;
63
64     return 0;
65
66
67
68
69 }
```

source: ...\\Modul 3\\Labaratorium 3\\Zadanie 2\\zadanie_2.cpp

Konstruktor&Destruktor:

```
zadanie_2.cpp x
zadanie_2 Punkt setName(const string _name)
70
71 Punkt::Punkt() {
72     x = 0; y = 0; name = "NONAME";
73     cout << "Konstruktor" << endl;
74 }
75
76
77
78 Punkt::~Punkt() {
79     cout << "Destruktor" << endl;
80 }
81
82
83 }
```

Wyjście:



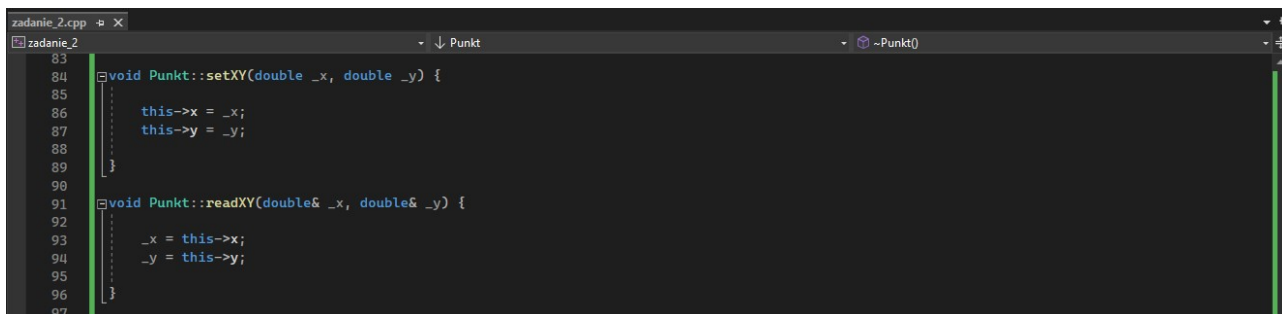
```
Microsoft Visual Studio Debug Console
Konstruktor
Konstruktor
Konstruktor
Nazwa: NONAME
W punkcie NONAME:
x = 0
y = 0
Podaj nazwe: zxc
Nazwa: zxc
Podaj x, y:
x = 10
y = 20
W punkcie zxc:
x = 10
y = 20
Destruktor
Destruktor
Destruktor
C:\Users\qawith\Desktop\1\CPP\Semestr 2\PPK IBrzezek\Modul 3\Labaratorium 3\Zadanie 2\x64\Debug\zadanie_2.exe (process 15512) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

W tym przypadku mamy trzy konstruktory przeciążone. Pierwszy nie przyjmuje żadnych parametrów i ustawia wartości domyślne dla zmiennych składowych. Drugi przyjmuje dwie wartości typu double, które ustawiają zmienne składowe x i y, a nazwa punktu jest ustawiana jako "NONAME". Trzeci konstruktor przyjmuje trzy parametry: dwie wartości typu double dla zmiennych składowych x i y oraz const referencję do obiektu string dla nazwy punktu.

Omówić słowo kluczowe this na przykładzie inicjalizacji cech obiektu argumentami konstruktora posiadającymi tą samą nazwę jak cechy. Zastanowić się jak tworzone i przechowywane są obiekty. Czy obiekty utworzone tak jak każda inna zmienna są w trakcie pracy kasowane gdy programista ich nie używa?

Słowo kluczowe **this** w odnosi się do wskaźnika na obiekt, na rzecz którego została wywołana metoda klasy.

Przykładowa implementacja w tym programie:



```
zadanie_2.cpp
void Punkt::setXY(double _x, double _y) {
    this->x = _x;
    this->y = _y;
}
void Punkt::readXY(double& _x, double& _y) {
    _x = this->x;
    _y = this->y;
}
```

source: ...\\Modul 3\\Labaratorium 3\\Zadanie 2\\zadanie_2.cpp

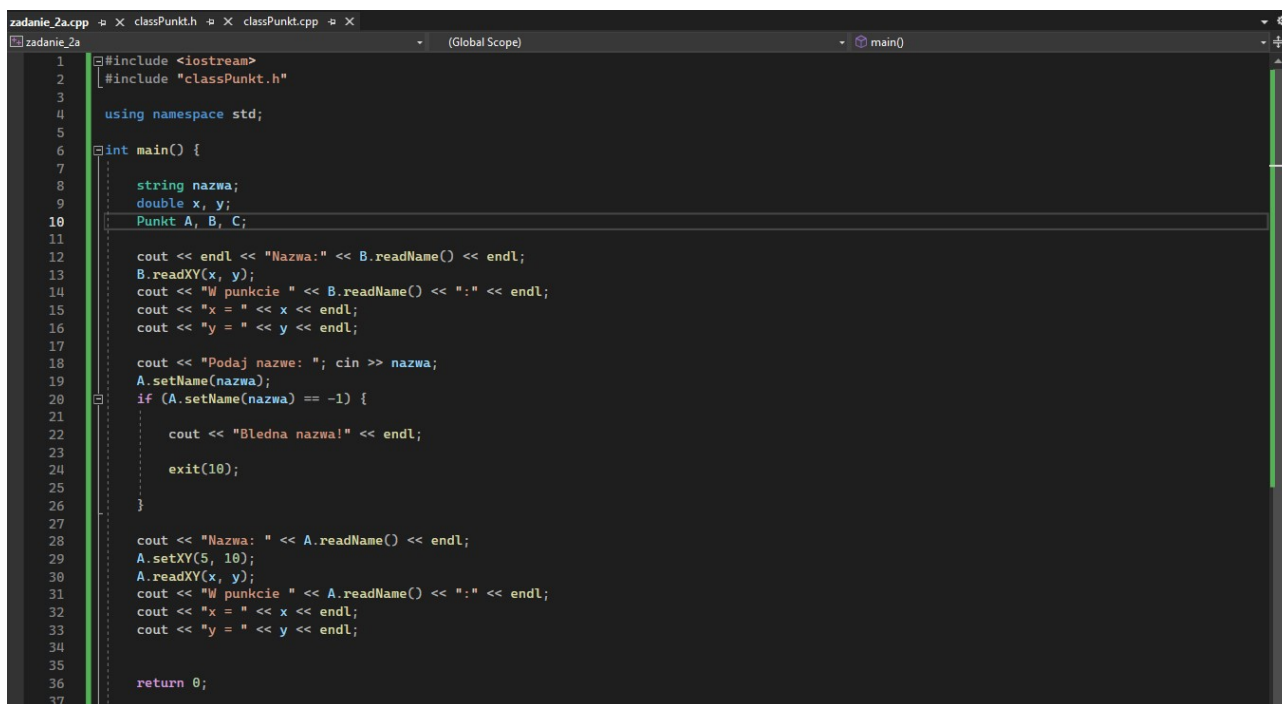
Obiekty w C++ są tworzone za pomocą konstruktora klasy i przechowywane w pamięci do momentu, gdy zostaną usunięte przez programistę lub do końca działania programu.

Zadanie 2a

Powyższy kod podzielić na plik nagłówkowy oraz na plik modułu z kodem metod.

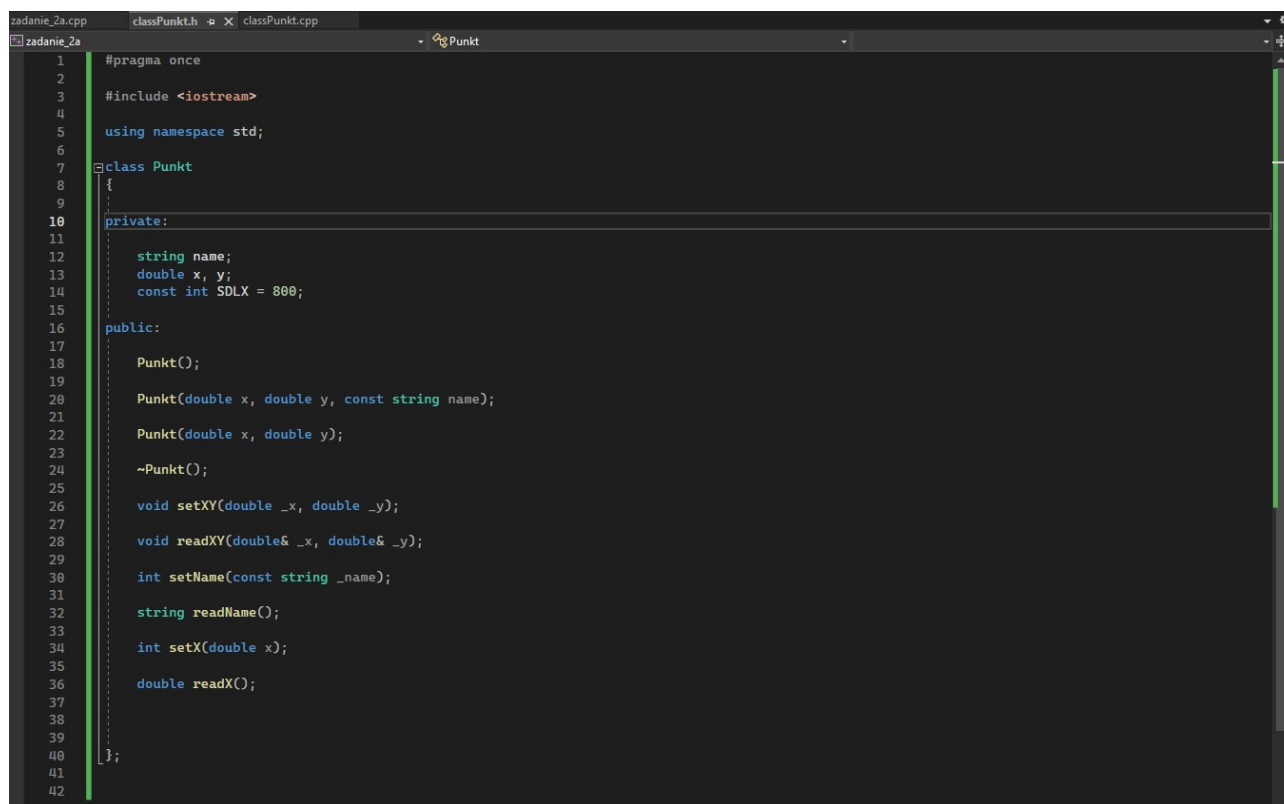
Rozwiązanie zadania 2a

Plik zadanie_2.cpp



```
1 #include <iostream>
2 #include "classPunkt.h"
3
4 using namespace std;
5
6 int main() {
7
8     string nazwa;
9     double x, y;
10    Punkt A, B, C;
11
12    cout << endl << "Nazwa: " << B.readName() << endl;
13    B.readXY(x, y);
14    cout << "W punkcie " << B.readName() << ":" << endl;
15    cout << "x = " << x << endl;
16    cout << "y = " << y << endl;
17
18    cout << "Podaj nazwe: "; cin >> nazwa;
19    A.setName(nazwa);
20    if (A.setName(nazwa) == -1) {
21
22        cout << "Bledna nazwa!" << endl;
23
24        exit(10);
25    }
26
27    cout << "Nazwa: " << A.readName() << endl;
28    A.setXY(5, 10);
29    A.readXY(x, y);
30    cout << "W punkcie " << A.readName() << ":" << endl;
31    cout << "x = " << x << endl;
32    cout << "y = " << y << endl;
33
34    return 0;
35
36
37 }
```

source: ...\\Modul 3\\Labaratorium 3\\Zadanie 2a\\zadanie_.cpp

Plik nagłówkowy classPunkt.h:

```
1  #pragma once
2
3  #include <iostream>
4
5  using namespace std;
6
7  class Punkt
8  {
9
10 private:
11
12     string name;
13     double x, y;
14     const int SDLX = 800;
15
16 public:
17
18     Punkt();
19
20     Punkt(double x, double y, const string name);
21
22     Punkt(double x, double y);
23
24     ~Punkt();
25
26     void setXY(double _x, double _y);
27
28     void readXY(double& _x, double& _y);
29
30     int setName(const string _name);
31
32     string readName();
33
34     int setX(double x);
35
36     double readX();
37
38
39
40 };
41
42
```

source: ...\\Modul 3\\Labaratorium 3\\Zadanie 2a\\classPunkt.h

Plik modułu ciała classPunkt.cpp:

```
zadanie_2a.cpp  classPunkt.h  classPunkt.cpp  X
zadanie_2a
1  #include "classPunkt.h"
2  #include <iostream>
3
4  using namespace std;
5
6  Punkt::Punkt() {
7
8      x = 0; y = 0; name = "NONAME";
9      cout << "Konstruktor" << endl;;
10 }
11
12
13 Punkt::Punkt(double x, double y, const string name) {
14
15 }
16
17
18 Punkt::Punkt(double x, double y) {
19
20 }
21
22
23
24
25 Punkt::~Punkt() {
26
27     cout << "Destruktor" << endl;
28 }
29
30
31 void Punkt::setXY(double _x, double _y) {
32
33     this->x = _x;
34     this->y = _y;
35 }
36
37
38 void Punkt::readXY(double& _x, double& _y) {
39
40     _x = this->x;
41     _y = this->y;
42 }
43
44
45 int Punkt::setName(const string _name) {
46
47     if (_name.length() < 3) {
48
49         return -1;
50     }
51
52     this->name = _name;
53     return 0;
54 }
55
56
57
58
59
60
61
62
63
64
65 int Punkt::setX(double x_) {
66
67     if (x > SDLX) {
68
69         return -1;
70     }
71
72     this->x = x;
73     return 0;
74 }
75
76
77
78 double Punkt::readX() {
79
80     return x;
81 }
82 }
```

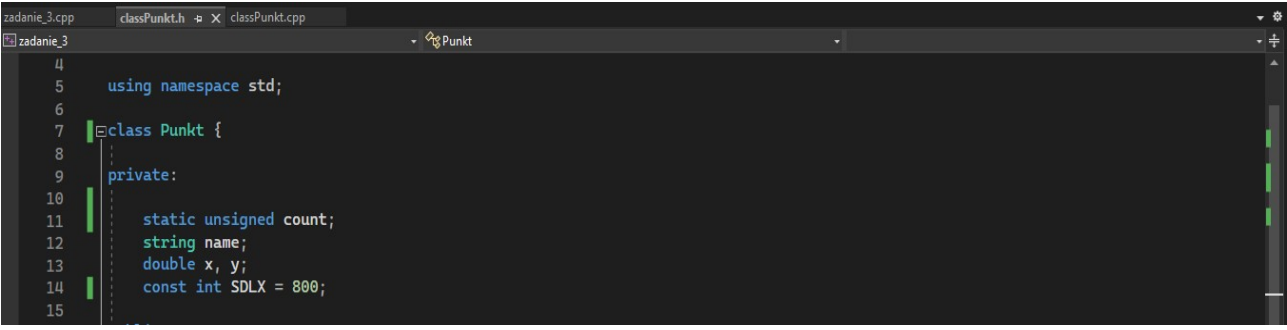
source: ...\\Modul 3\\Labaratorium 3\\Zadanie 2a\\classPunkt.cpp

Zadanie 3

Program rozbudować o zmienną statyczną w klasie i utworzyć kilka obiektów. Do konstruktora dodać zwiększanie takiej zmiennej o 1. Zaobserwować i omówić działanie podczas tworzenia kolejnych obiektów. Jak działa zmienna statyczna? Jak można odczytać jej wartości po utworzeniu kolejnych np trzech obiektów? Przeprowadzić test licznika kiedy nie jest on zmienną statyczną.

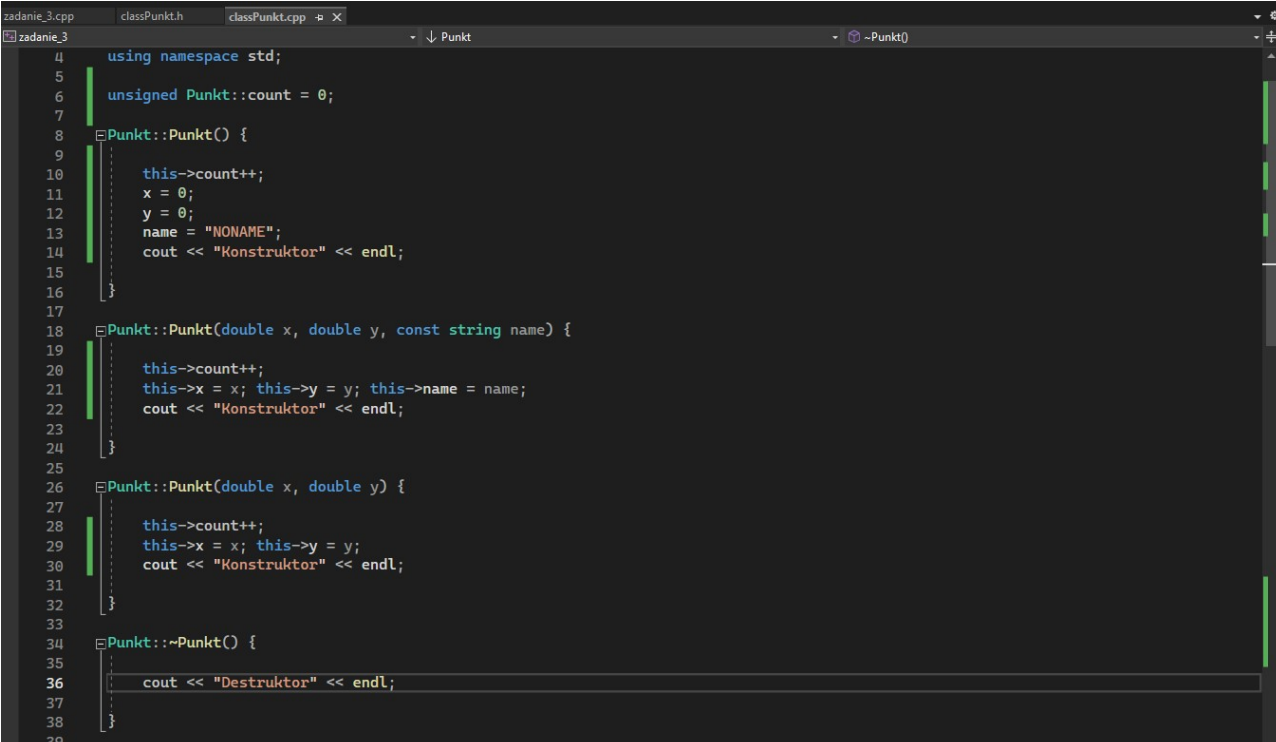
Rozwiązanie zadania 3

Program rozbudować o zmienną statyczną w klasie i utworzyć kilka obiektów. Do konstruktora dodać zwiększanie takiej zmiennej o 1. Zaobserwować i omówić działanie podczas tworzenia kolejnych obiektów.

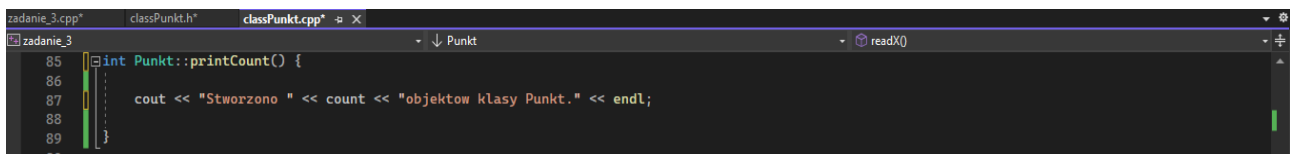


```
1  zadanie_3.cpp  classPunkt.h  classPunkt.cpp
2  zadanie_3
3  Punkt
4  4
5  5 using namespace std;
6  6
7  7 class Punkt {
8  8
9  9 private:
10 10
11 11 static unsigned count;
12 12 string name;
13 13 double x, y;
14 14 const int SDLX = 800;
15 15
```

source: ...\\Modul 3\\Labaratorium 3\\Zadanie 3\\classPunkt.h



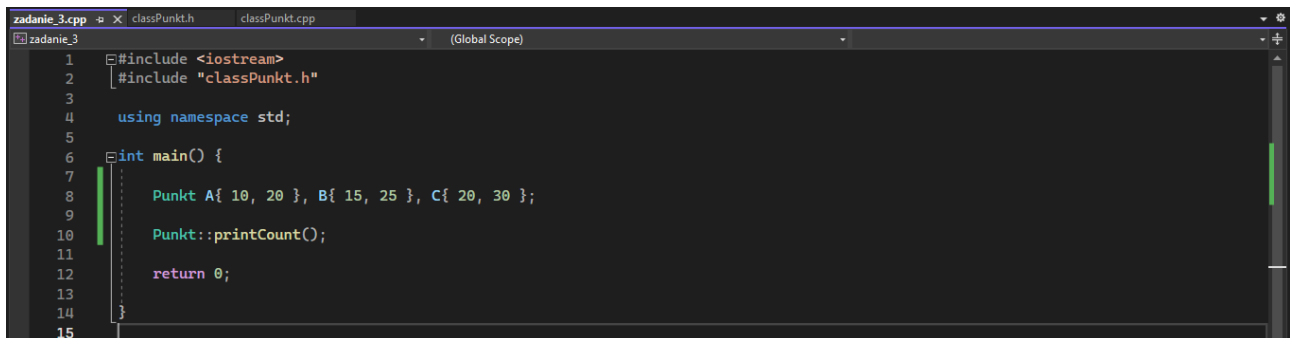
```
1  zadanie_3.cpp  classPunkt.h  classPunkt.cpp
2  zadanie_3
3  Punkt
4  ~Punkt()
5  4 using namespace std;
6  5 unsigned Punkt::count = 0;
7  6
8  7 Punkt::Punkt() {
9  8
10 9 this->count++;
11 10 x = 0;
12 11 y = 0;
13 12 name = "NONAME";
14 13 cout << "Konstruktor" << endl;
15 14
16 15 }
17 16
18 17 Punkt::Punkt(double x, double y, const string name) {
19 18
20 19 this->count++;
21 20 this->x = x; this->y = y; this->name = name;
22 21 cout << "Konstruktor" << endl;
23 22
24 23 }
25 24
26 25 Punkt::Punkt(double x, double y) {
27 26
28 27 this->count++;
29 28 this->x = x; this->y = y;
30 29 cout << "Konstruktor" << endl;
31 30
32 31 }
33 32
34 33 Punkt::~Punkt() {
35 34
36 35 cout << "Destruktor" << endl;
37 36
38 37 }
39 38
40 39
```



```
zadanie_3.cpp* classPunkt.h* classPunkt.cpp* X
zadanie_3
85 int Punkt::printCount() {
86
87     cout << "Stworzono " << count << " obiektow klasy Punkt." << endl;
88
89 }
```

source: ...\\Modul 3\\Labaratorium 3\\Zadanie 3\\classPunkt.cpp

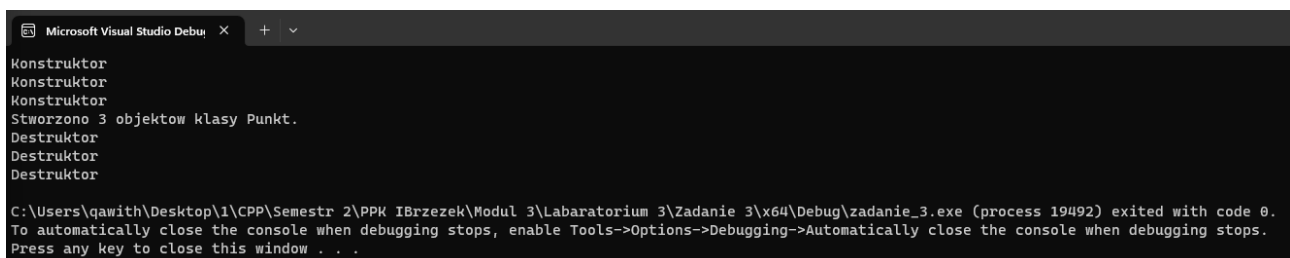
Main:



```
zadanie_3.cpp classPunkt.h classPunkt.cpp
zadanie_3 (Global Scope)
1 #include <iostream>
2 #include "classPunkt.h"
3
4 using namespace std;
5
6 int main() {
7
8     Punkt A{ 10, 20 }, B{ 15, 25 }, C{ 20, 30 };
9
10    Punkt::printCount();
11
12    return 0;
13
14 }
15
```

source: ...\\Modul 3\\Labaratorium 3\\Zadanie 3\\zadanie_3.cpp

Wyjście:

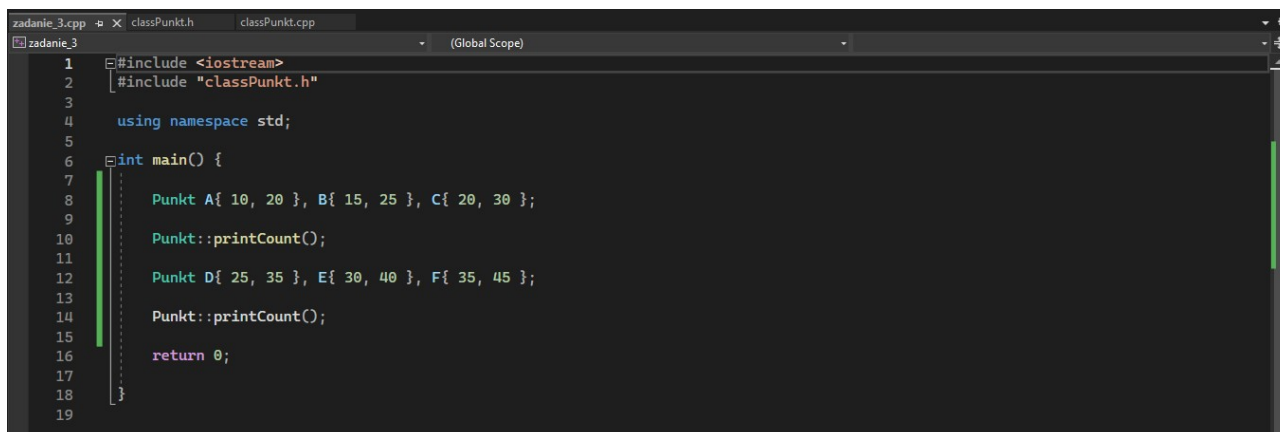


```
Microsoft Visual Studio Debu X + v
Konstruktor
Konstruktor
Konstruktor
Stworzono 3 obiektow klasy Punkt.
Destruktor
Destruktor
Destruktor
C:\Users\qawith\Desktop\1\CPP\Semestr 2\PPK IBrzezek\Modul 3\Labaratorium 3\Zadanie 3\x64\Debug\zadanie_3.exe (process 19492) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

Jak działa zmienna statyczna?

Zmienna statyczna jest wspólna dla wszystkich obiektów klasy. W tym przypadku zostaną utworzone 3 obiekty klasy **Punkt**, tzn. że **count** = 3.

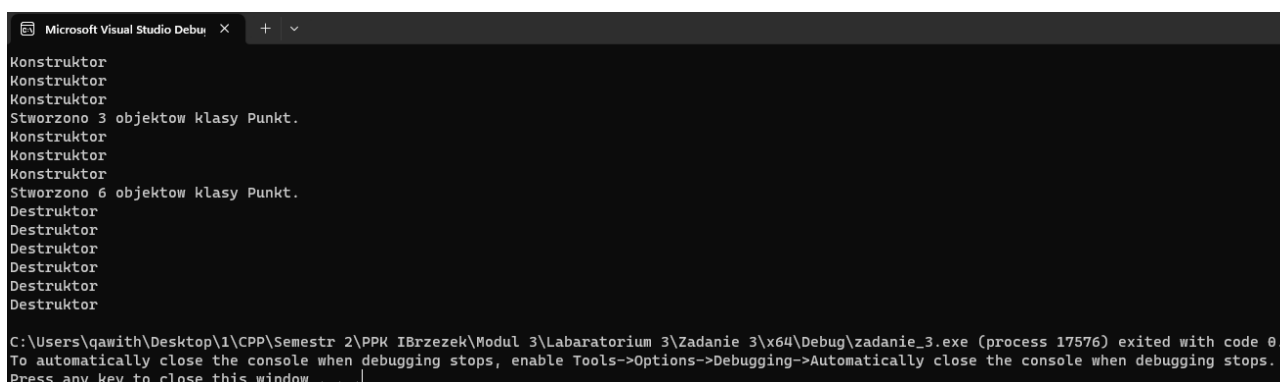
Jakie można odczytać jej wartości po utworzeniu kolejnych np trzech obiektów?



```
1 #include <iostream>
2 #include "classPunkt.h"
3
4 using namespace std;
5
6 int main() {
7
8     Punkt A{ 10, 20 }, B{ 15, 25 }, C{ 20, 30 };
9
10    Punkt::printCount();
11
12    Punkt D{ 25, 35 }, E{ 30, 40 }, F{ 35, 45 };
13
14    Punkt::printCount();
15
16    return 0;
17
18 }
19
```

source: ...\\Modul 3\\Labaratorium 3\\Zadanie 3\\zadanie_3.cpp

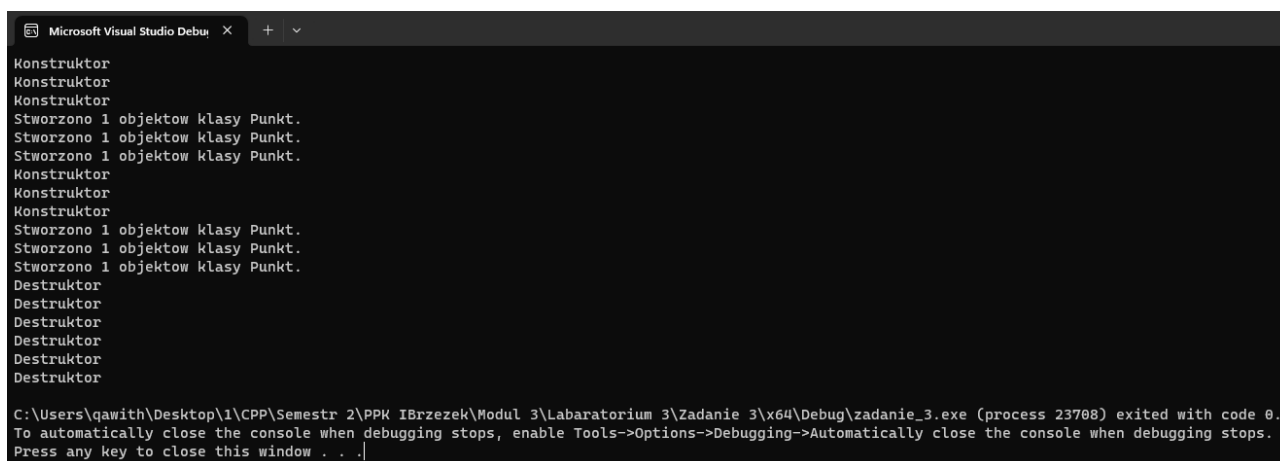
Wyjście:



```
Konstruktor
Konstruktor
Konstruktor
Stworzono 3 obiektow klasy Punkt.
Konstruktor
Konstruktor
Konstruktor
Stworzono 6 obiektow klasy Punkt.
Destruktor
Destruktor
Destruktor
Destruktor
Destruktor
Destruktor

C:\Users\qawith\Desktop\1\CPP\Semestr 2\PPK IBrzezek\Modul 3\Labaratorium 3\Zadanie 3\x64\Debug\zadanie_3.exe (process 17576) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

Przeprowadzić test licznika kiedy nie jest on zmienną statyczną.



```
Konstruktor
Konstruktor
Konstruktor
Stworzono 1 obiektow klasy Punkt.
Stworzono 1 obiektow klasy Punkt.
Stworzono 1 obiektow klasy Punkt.
Konstruktor
Konstruktor
Konstruktor
Stworzono 1 obiektow klasy Punkt.
Stworzono 1 obiektow klasy Punkt.
Stworzono 1 obiektow klasy Punkt.
Destruktor
Destruktor
Destruktor
Destruktor
Destruktor
Destruktor

C:\Users\qawith\Desktop\1\CPP\Semestr 2\PPK IBrzezek\Modul 3\Labaratorium 3\Zadanie 3\x64\Debug\zadanie_3.exe (process 23708) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

Kiedy zmienna nie jest statyczną – ona nie jest wspólna dla wszystkich obiektów, tzn. że wartość

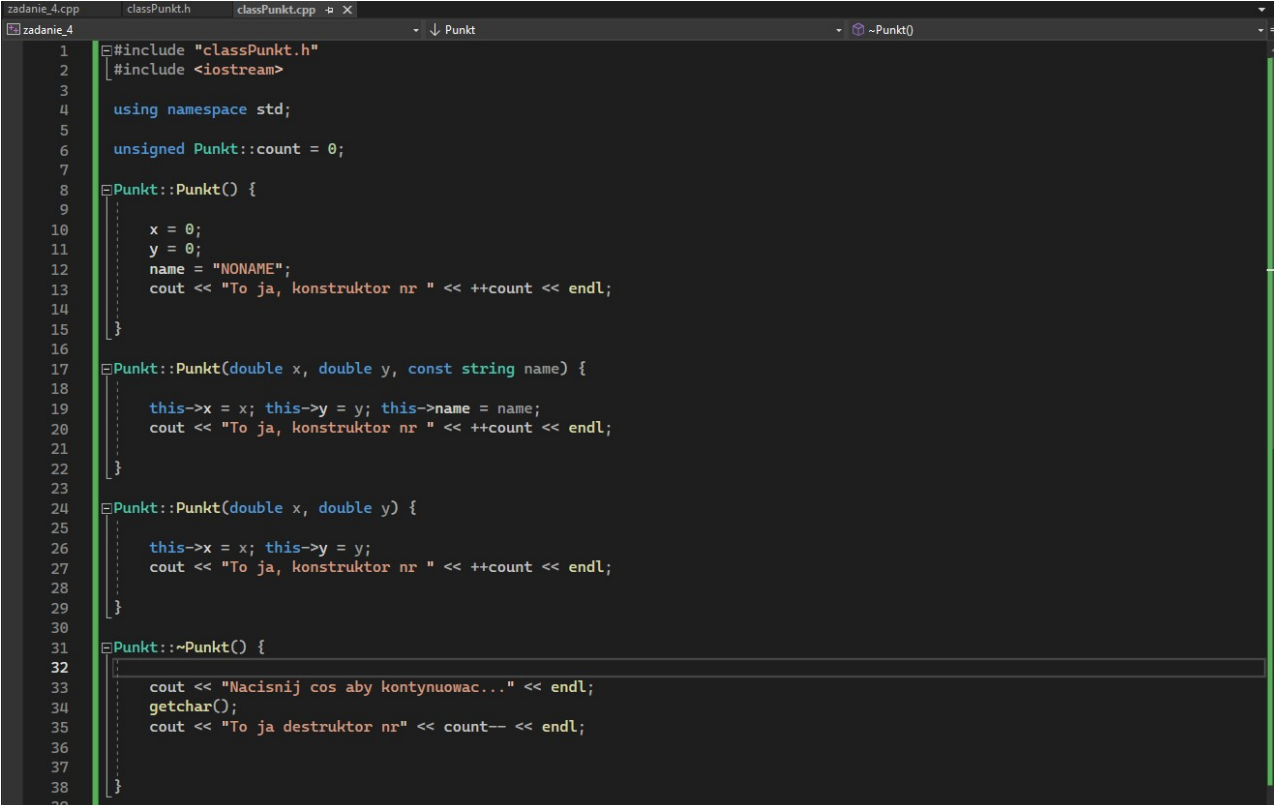
zmiennej **count** będzie równa 1.

Zadanie 4

Rozbudować powyższy program o dynamiczne tworzenie i niszczenie obiektów. Dodać do konstruktora wyświetlanie tekstu: To ja, konstruktor nr .. podającą wartość zmiennej statycznej po inkrementacji a tym samym numer obiektu. Dodać do klasy destruktor wyświetlający tekst "To ja destruktor" oraz zmniejszający o 1 zmienną statyczną z poprzedniego przykładu. Utworzyć kilka dynamicznych obiektów i kolejno je usunąć. Usuwanie obiektów rozdzielić zatrzymaniem akcji programu z wyświetleniem informacji typu "Nacisnij cos aby kontynuowac...". Kiedy pojawi się napis konstruktora? Jak są zatem tworzone obiekty? Gdzie i kiedy są tworzone statyczne a gdzie i kiedy dynamiczne?

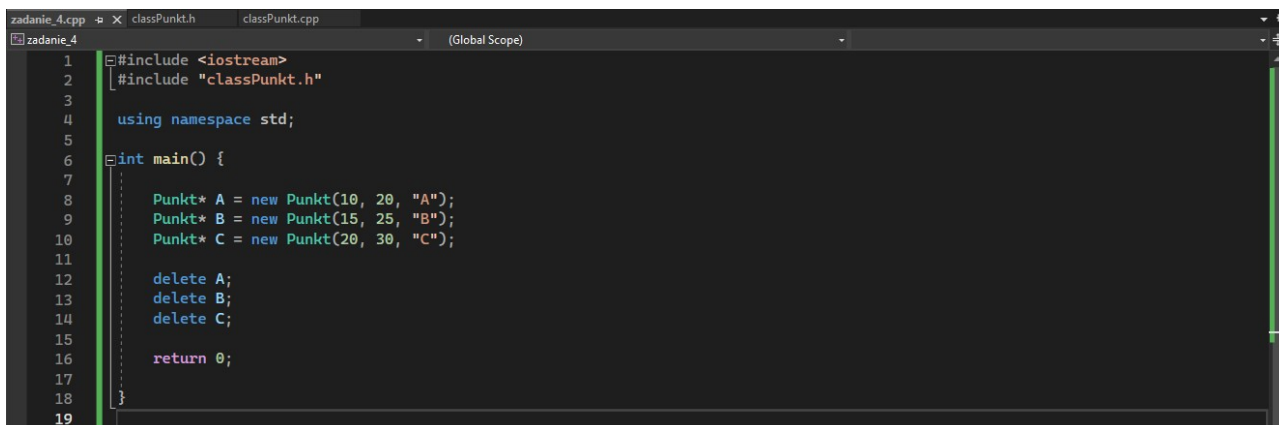
Rozwiązanie zadania 4

Rozbudować powyższy program o dynamiczne tworzenie i niszczenie obiektów. Dodać do konstruktora wyświetlanie tekstu: To ja, konstruktor nr .. podającą wartość zmiennej statycznej po inkrementacji a tym samym numer obiektu. Dodać do klasy destruktor wyświetlający tekst "To ja destruktor" oraz zmniejszający o 1 zmienną statyczną z poprzedniego przykładu. Utworzyć kilka dynamicznych obiektów i kolejno je usunąć. Usuwanie obiektów rozdzielić zatrzymaniem akcji programu z wyświetleniem informacji typu "Nacisnij cos aby kontynuowac...".



```
1 #include "classPunkt.h"
2 #include <iostream>
3
4 using namespace std;
5
6 unsigned Punkt::count = 0;
7
8 Punkt::Punkt() {
9
10     x = 0;
11     y = 0;
12     name = "NONAME";
13     cout << "To ja, konstruktor nr " << ++count << endl;
14 }
15
16
17 Punkt::Punkt(double x, double y, const string name) {
18
19     this->x = x; this->y = y; this->name = name;
20     cout << "To ja, konstruktor nr " << ++count << endl;
21 }
22
23
24 Punkt::Punkt(double x, double y) {
25
26     this->x = x; this->y = y;
27     cout << "To ja, konstruktor nr " << ++count << endl;
28 }
29
30
31 Punkt::~Punkt() {
32
33     cout << "Nacisnij cos aby kontynuowac..." << endl;
34     getchar();
35     cout << "To ja destruktor nr" << count-- << endl;
36 }
37
38
39 }
```

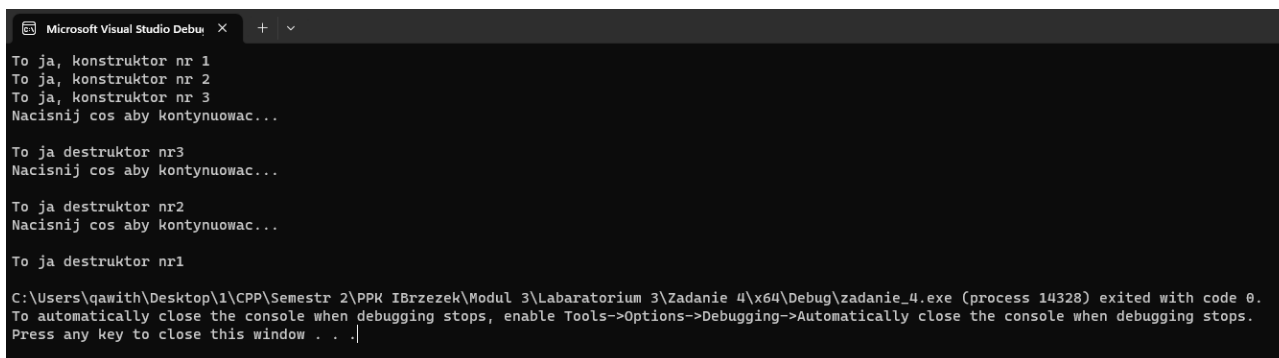
source: ...\\Modul 3\\Labaratorium 3\\Zadanie 4\\classPunkt.cpp



```
1 #include <iostream>
2 #include "classPunkt.h"
3
4 using namespace std;
5
6 int main() {
7
8     Punkt* A = new Punkt(10, 20, "A");
9     Punkt* B = new Punkt(15, 25, "B");
10    Punkt* C = new Punkt(20, 30, "C");
11
12    delete A;
13    delete B;
14    delete C;
15
16    return 0;
17 }
18
19
```

source: ...\\Modul 3\\Labaratorium 3\\Zadanie 4\\zadanie_4.cpp

Wyjście:



```
Microsoft Visual Studio Debug
To ja, konstruktor nr 1
To ja, konstruktor nr 2
To ja, konstruktor nr 3
Nacisnij cos aby kontynuowac...

To ja destruktor nr3
Nacisnij cos aby kontynuowac...

To ja destruktor nr2
Nacisnij cos aby kontynuowac...

To ja destruktor nr1

C:\\Users\\qawith\\Desktop\\1\\CPP\\Semestr 2\\PPK IBrzezek\\Modul 3\\Labaratorium 3\\Zadanie 4\\x64\\Debug\\zadanie_4.exe (process 14328) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .|
```

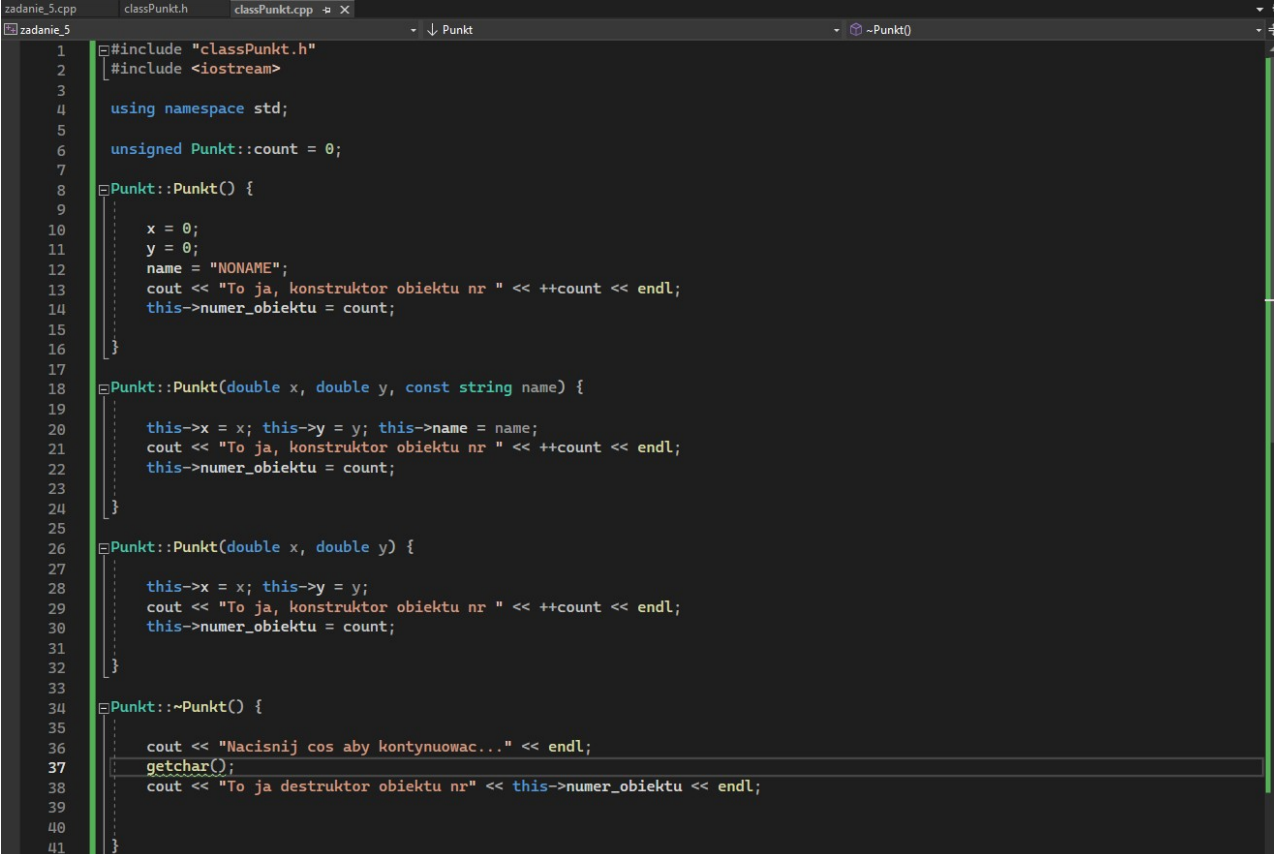
Kiedy pojawi się napis konstruktora? Jak są zatem tworzone obiekty? Gdzie i kiedy są tworzone statyczne a gdzie i kiedy dynamiczne?

Napis pojawi się od razu jak będzie stworzony nowy obiekt klasy **Punkt**. W tym programie obiekty są tworzone dynamicznie, tzn. z użyciem **new**. Statyczne obiekty są tworzone w momencie uruchomienia programu, jeszcze przed rozpoczęciem działania funkcji **main()**. Natomiast dynamiczne obiekty są tworzone w trakcie działania programu, za pomocą operatora **new**. Zawsze muszą być usuwane ręcznie, za pomocą operatora **delete**.

Zadanie 5

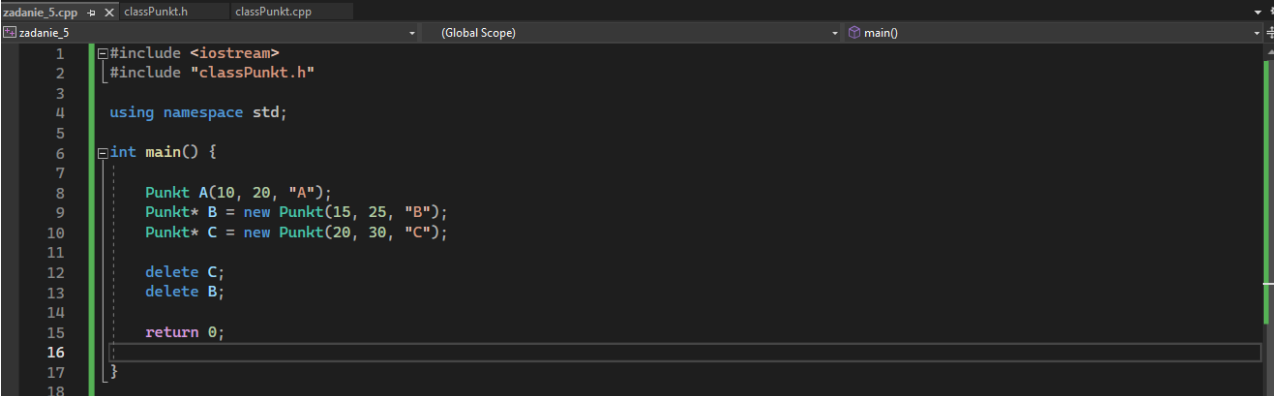
Do obiektu dodać cechę numerObiektu. Do tej cechy konstruktor ma wpisywać numer zmiennej statycznej "licznik" podczas tworzenia obiektu. Utworzyć trzy obiekty (statyczny i dwa dynamiczne) i kolejno usunąć oba dynamiczne ale w innej kolejności jak były tworzone. Dodać do destruktora wyświetlanie informacji o numerze usuwanego obiektu.

Rozwiązanie zadania 5

A screenshot of a code editor showing the implementation of the Punkt class in classPunkt.cpp. The code includes necessary headers, uses the std namespace, and initializes a static count variable. It defines three constructors: a default constructor, a constructor taking x, y, and name, and a constructor taking x and y. The destructor prints the object's number before deleting it. Line numbers 1 through 41 are visible on the left.

```
1 #include "classPunkt.h"
2 #include <iostream>
3
4 using namespace std;
5
6 unsigned Punkt::count = 0;
7
8 Punkt::Punkt() {
9
10     x = 0;
11     y = 0;
12     name = "NONAME";
13     cout << "To ja, konstruktor obiektu nr " << ++count << endl;
14     this->numer_obiektu = count;
15 }
16
17
18 Punkt::Punkt(double x, double y, const string name) {
19
20     this->x = x; this->y = y; this->name = name;
21     cout << "To ja, konstruktor obiektu nr " << ++count << endl;
22     this->numer_obiektu = count;
23 }
24
25
26 Punkt::Punkt(double x, double y) {
27
28     this->x = x; this->y = y;
29     cout << "To ja, konstruktor obiektu nr " << ++count << endl;
30     this->numer_obiektu = count;
31 }
32
33
34 Punkt::~~Punkt() {
35
36     cout << "Nacisnij cos aby kontynuowac..." << endl;
37     getchar();
38     cout << "To ja destruktor obiektu nr" << this->numer_obiektu << endl;
39 }
40
41 }
```

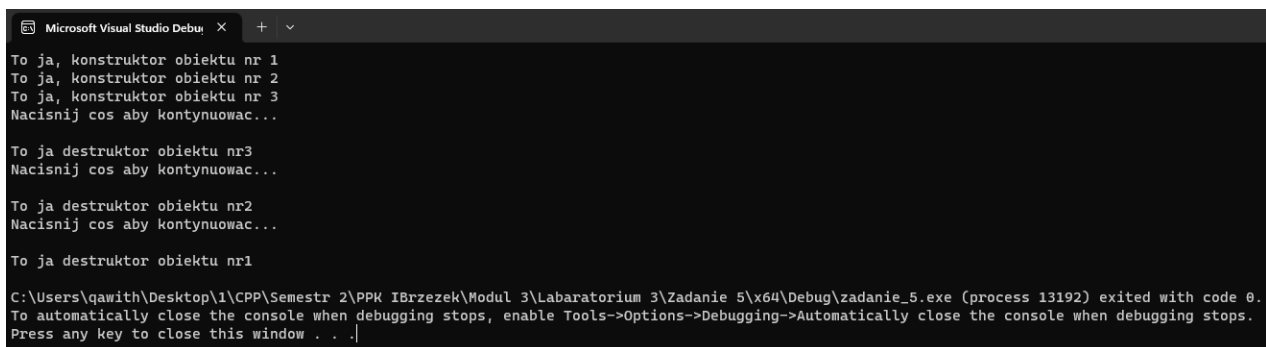
source: ...\\Modul 3\\Labaratorium 3\\Zadanie 5\\classPunkt.cpp

A screenshot of a code editor showing the main function in zadanie_5.cpp. It includes the iostream and classPunkt.h headers, uses the std namespace, and defines a main function that creates three objects: A (static), B (dynamic), and C (dynamic). It then deletes C and B in that order and returns 0. Line numbers 1 through 18 are visible on the left.

```
1 #include <iostream>
2 #include "classPunkt.h"
3
4 using namespace std;
5
6 int main() {
7
8     Punkt A(10, 20, "A");
9     Punkt* B = new Punkt(15, 25, "B");
10    Punkt* C = new Punkt(20, 30, "C");
11
12    delete C;
13    delete B;
14
15    return 0;
16 }
17
18 }
```

source: ...\\Modul 3\\Labaratorium 3\\Zadanie 5\\zadanie_5.cpp

Wyjście:



The screenshot shows a Visual Studio Debug Console window with a dark background. The title bar reads 'Microsoft Visual Studio Debug' with a close button and window controls. The output text is as follows:

```
To ja, konstruktor obiektu nr 1
To ja, konstruktor obiektu nr 2
To ja, konstruktor obiektu nr 3
Nacisnij cos aby kontynuowac...

To ja destruktor obiektu nr3
Nacisnij cos aby kontynuowac...

To ja destruktor obiektu nr2
Nacisnij cos aby kontynuowac...

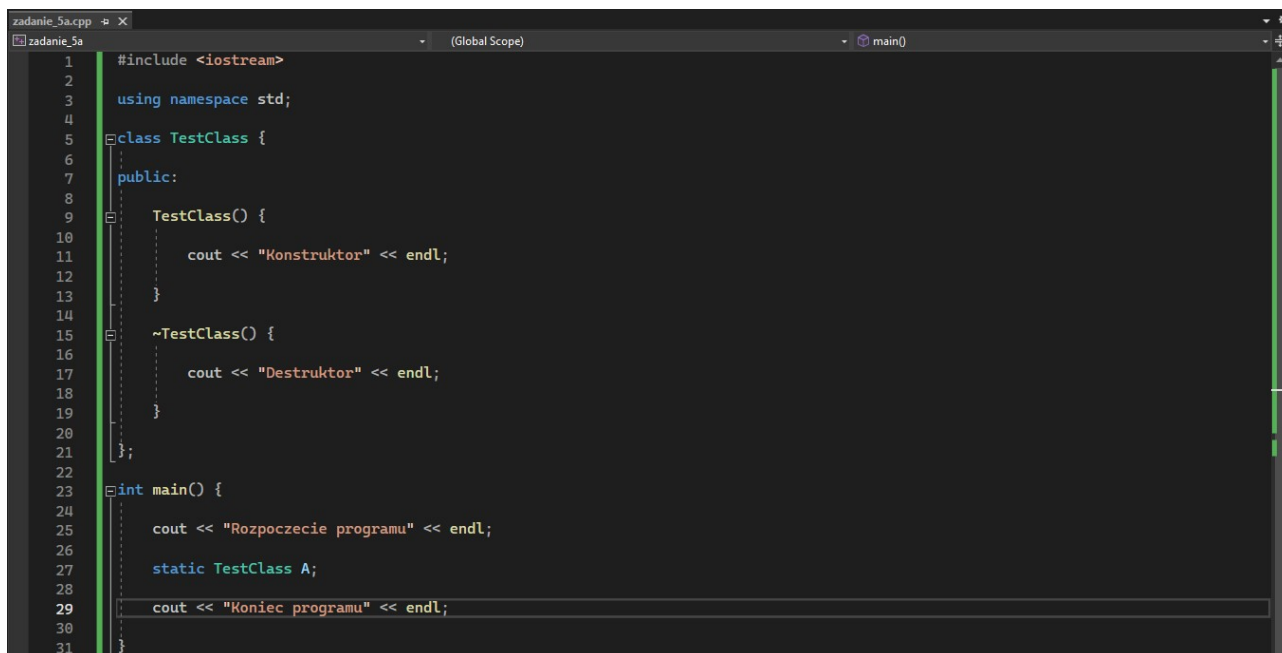
To ja destruktor obiektu nr1

C:\Users\qawith\Desktop\1\CPP\Semestr 2\PPK IBrzezek\Modul 3\Labaratorium 3\Zadanie 5\x64\Debug\zadanie_5.exe (process 13192) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .|
```

Zadanie 5a

Sprawdzić jak i kiedy niszczony jest obiekt statyczny (chodzi o taki, który nie jest związany ze zmienną dynamiczną). Test przeprowadzić na bazie bloku kodu i zawartego w nim obiektu.

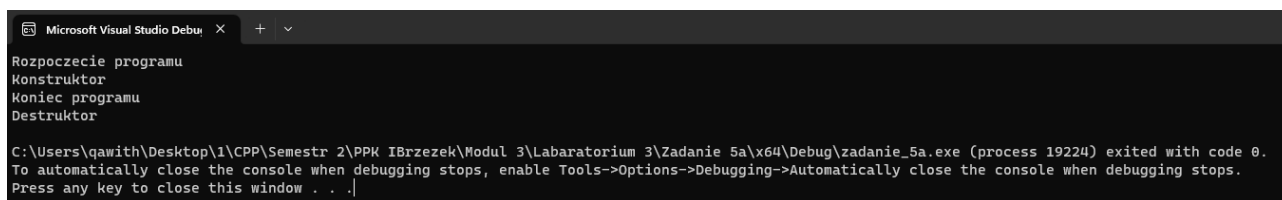
Rozwiązanie zadania 5a



```
1 #include <iostream>
2
3 using namespace std;
4
5 class TestClass {
6
7 public:
8
9     TestClass() {
10         cout << "Konstruktor" << endl;
11     }
12
13     ~TestClass() {
14         cout << "Destruktor" << endl;
15     }
16 };
17
18 int main() {
19     cout << "Rozpoczecie programu" << endl;
20
21     static TestClass A;
22
23     cout << "Koniec programu" << endl;
24 }
```

source: ...\\Modul 3\\Labaratorium 3\\Zadanie 5a\\zadanie_5a.cpp

Wyjście:



```
Microsoft Visual Studio Debug Console
Rozpoczecie programu
Konstruktor
Koniec programu
Destruktor

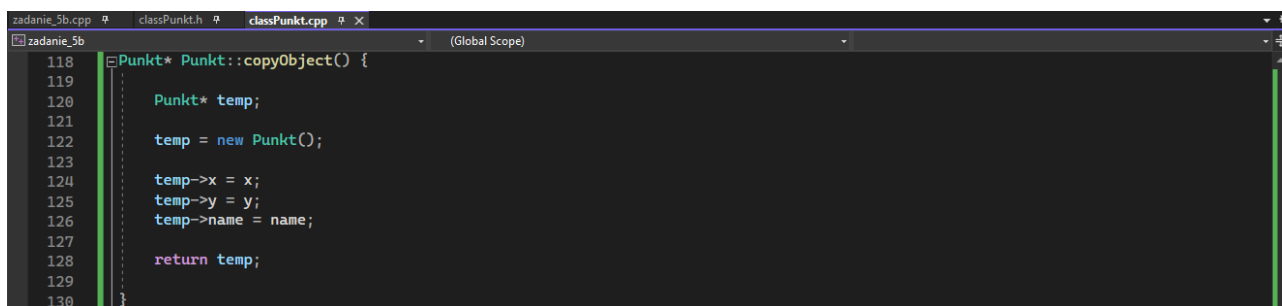
C:\Users\qawith\Desktop\1\CPP\Semestr 2\PPK IBrzezek\Modul 3\Labaratorium 3\Zadanie 5a\x64\Debug\zadanie_5a.exe (process 19224) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

Zadanie 5b

Korzystając z klasy Punkt utworzyć zmienne obiektowe p1 i p2 (wskaźnikowe) bez ich inicjacji. Pierwszą zmienną zainicjalizować dowolnymi danymi. Wyświetlić cechy obu obiektów. Skopiować pierwszy obiekt do drugiego. W pierwszym obiekcie zmienić współrzędne i kolejno wyświetlić cechy obu obiektów. Jaki jest efekt? Z czego on wynika? Opracować metodę poprawnego kopiowania obiektu p1 do p2. Co to znaczy "kopiowanie obiektu"? Co jest w powyższym przykładzie kopiowane? Obiekt, zawartość, a może jeszcze coś innego? Co zatem powinno być kopiowane?

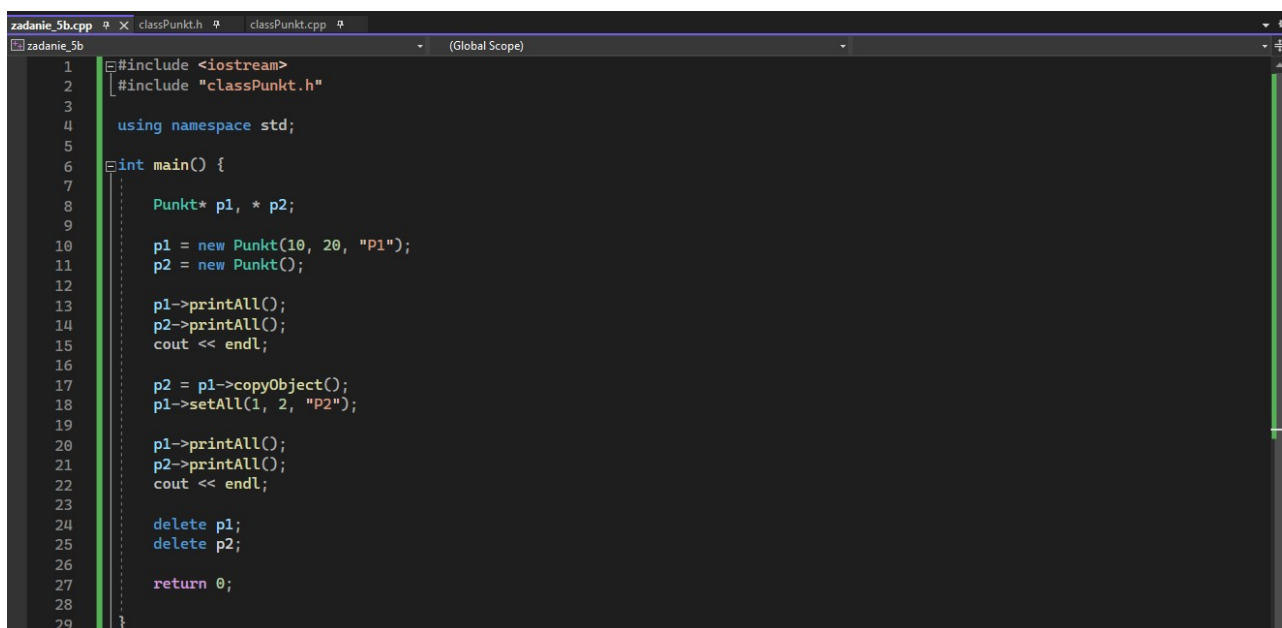
Rozwiązanie zadania 5b

Korzystając z klasy Punkt utworzyć zmienne obiektowe p1 i p2 (wskaźnikowe) bez ich inicjacji. Pierwszą zmienną zainicjalizować dowolnymi danymi. Wyświetlić cechy obu obiektów. Skopiować pierwszy obiekt do drugiego. W pierwszym obiekcie zmienić współrzędne i kolejno wyświetlić cechy obu obiektów.



```
118 Punkt* Punkt::copyObject() {
119
120     Punkt* temp;
121
122     temp = new Punkt();
123
124     temp->x = x;
125     temp->y = y;
126     temp->name = name;
127
128     return temp;
129 }
130 }
```

source: ...\\Modul 3\\Labaratorium 3\\Zadanie 5b\\classPunkt.cpp



```
1 #include <iostream>
2 #include "classPunkt.h"
3
4 using namespace std;
5
6 int main() {
7
8     Punkt* p1, * p2;
9
10    p1 = new Punkt(10, 20, "p1");
11    p2 = new Punkt();
12
13    p1->printAll();
14    p2->printAll();
15    cout << endl;
16
17    p2 = p1->copyObject();
18    p1->setAll(1, 2, "p2");
19
20    p1->printAll();
21    p2->printAll();
22    cout << endl;
23
24    delete p1;
25    delete p2;
26
27    return 0;
28 }
29 }
```

source: ...\\Modul 3\\Labaratorium 3\\Zadanie 5b\\zadanie_5b.cpp

Wyjście:

```
Microsoft Visual Studio Debu X + v
To ja, konstruktor obiektu nr 1
To ja, konstruktor obiektu nr 2
W punkcie P1:
x = 10
y = 20
W punkcie empty:
x = 0
y = 0

To ja, konstruktor obiektu nr 3
W punkcie P2:
x = 1
y = 2
W punkcie P1:
x = 10
y = 20

Nacisnij cos aby kontynuowac...

To ja destruktor obiektu nr1
Nacisnij cos aby kontynuowac...

To ja destruktor obiektu nr3

C:\Users\qawith\Desktop\1\CPP\Semestr 2\PPK IBrzezek\Modul 3\Labaratorium 3\Zadanie 5b\x64\Debug\zadanie_5b.exe (process 13844) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .|
```

Jaki jest efekt? Z czego on wynika? Opracować metodę poprawnego kopiowania obiektu p1 do p2. Co to znaczy "kopiowanie obiektu"? Co jest w powyższym przykładzie kopiowane? Obiekt, zawartość, a może jeszcze coś innego? Co zatem powinno być kopiowane?

W wyniku kopiowania obiektu p1 do p2, oba wskaźniki wskazywały na ten sam obszar pamięci, co spowodowało, że zmiana jednego obiektu powodowała zmianę drugiego. Kopiowanie obiektu polega na utworzeniu nowego obiektu z taką samą zawartością jak oryginał. W powyższym przykładzie kopiowane są wartości atrybutów obiektów klasy Punkt, czyli wartości **x**, **y** i **name**. Powinny one być kopiowane, ponieważ to one określają stan obiektu.

Zadanie 5c

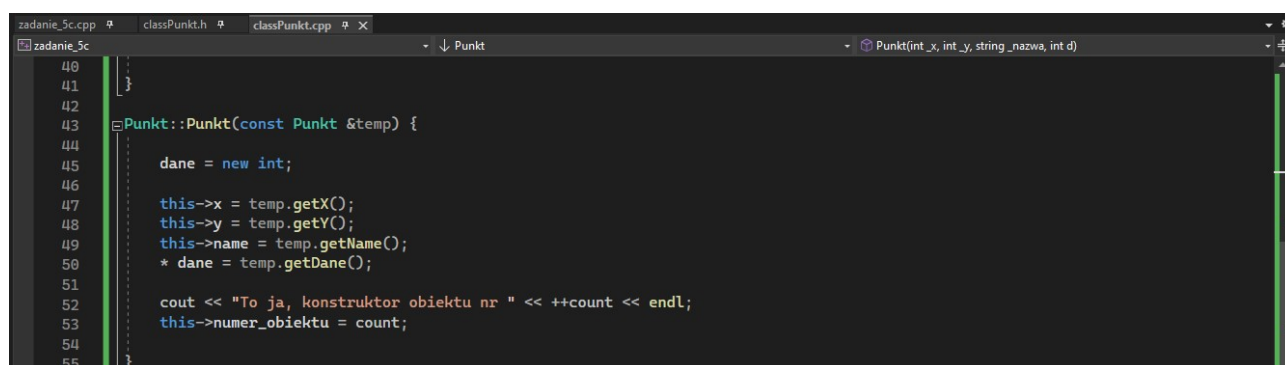
Konstruktor kopiujący.

Uruchamiany jest gdy obiekt jest kopiowany lub przekazywany do funkcji jako argument przez wartość (wtedy tworzona jest w funkcji kopia obiektu). Kiedy jednak w obiekcie znajdują się zmienne dynamiczne (wskaźniki) sprawa się komplikuje ponieważ taka kopia otrzymuje kopię wskaźnika, która wskazuje na ten sam obszar pamięci co oryginał. Modyfikacja zawartości jakiej zmiennej dynamicznej w "kopii" jest tak naprawdę modyfikacją w obiekcie oryginalnym i wspak.

Napisać program tworzący obiekt klasy A na bazie innego, istniejącego już obiektu tej samej klasy. Wykorzystać konstruktor kopiujący. Omówić dwa przypadki: kiedy w obiekcie brak zmiennych dynamicznych oraz kiedy w obiekcie znajduje się zmienna dynamiczna. Co w takim przypadku jest kopiowane? Jak to wykonać poprawnie?

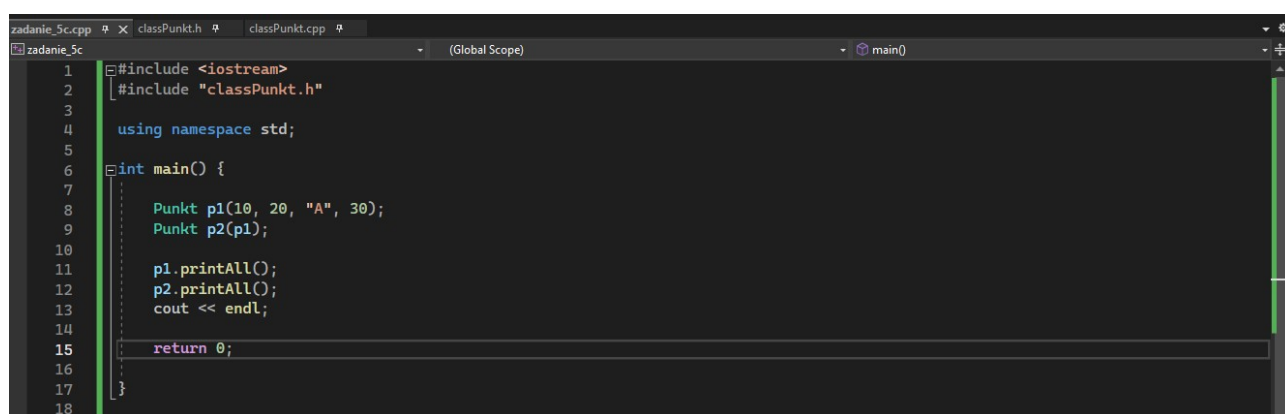
Rozwiązanie zadania 5c

Napisać program tworzący obiekt klasy A na bazie innego, istniejącego już obiektu tej samej klasy. Wykorzystać konstruktor kopiujący.



```
40  
41  
42  
43 Punkt::Punkt(const Punkt &temp) {  
44  
45     dane = new int;  
46  
47     this->x = temp.getX();  
48     this->y = temp.getY();  
49     this->nazwa = temp.getName();  
50     * dane = temp.getDane();  
51  
52     cout << "To ja, konstruktor obiektu nr " << ++count << endl;  
53     this->numer_obiektu = count;  
54  
55 }
```

source: ...\\Modul 3\\Labaratorium 3\\Zadanie 5c\\classPunkt.cpp



```
1 #include <iostream>  
2 #include "classPunkt.h"  
3  
4 using namespace std;  
5  
6 int main() {  
7  
8     Punkt p1(10, 20, "A", 30);  
9     Punkt p2(p1);  
10  
11     p1.printAll();  
12     p2.printAll();  
13     cout << endl;  
14  
15     return 0;  
16  
17 }  
18
```

source: ...\\Modul 3\\Labaratorium 3\\Zadanie 5c\\zadanie_5c.cpp

Wyjście:

```
Microsoft Visual Studio Debu X + v
Zwykly konstruktor obiektu: A
To ja, konstruktor obiektu nr 1
W punkcie A:
x = 10
y = 20
W punkcie A:
x = 10
y = 20

Nacisnij cos aby kontynuowac...

To ja destruktor obiektu nr1
Nacisnij cos aby kontynuowac...

To ja destruktor obiektu nr0

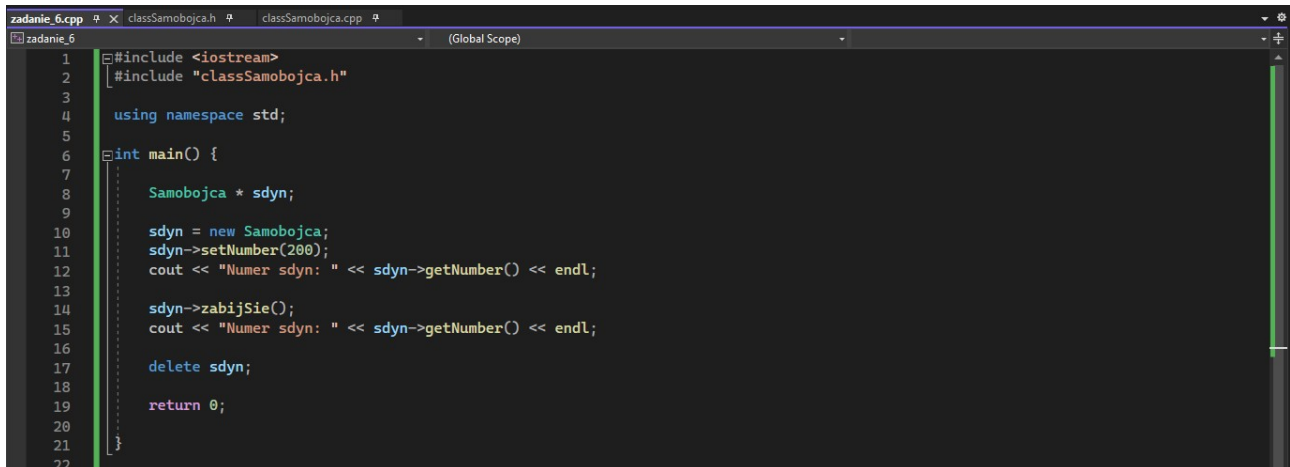
C:\Users\qawith\Desktop\1\CPP\Semestr 2\PPK IBrzezek\Modul 3\Labaratorium 3\Zadanie 5c\x64\Debug\zadanie_5c.exe (process 13480) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .|
```

Omówić dwa przypadki: kiedy w obiekcie brak zmiennych dynamicznych oraz kiedy w obiekcie znajduje się zmienna dynamiczna. Co w takim przypadku jest kopiowane? Jak to wykonać poprawnie?

W przypadku, gdy obiekt klasy nie posiada zmiennych dynamicznych, konstruktor kopiujący jest niepotrzebny i można skorzystać z domyślnego konstruktora kopiującego, który skopiuje pola obiektu. Jeśli jednak w klasie występują zmienne dynamiczne, to kopiowanie obiektu przez skopiowanie wskaźnika na dynamiczną pamięć nie działa poprawnie, ponieważ oba obiekty będą wskazywać na ten sam obszar pamięci. W takim przypadku trzeba przedefiniować konstruktor kopiujący, aby skopiował wartości zmiennych dynamicznych i utworzył nowy obszar pamięci na skopiowane dane.

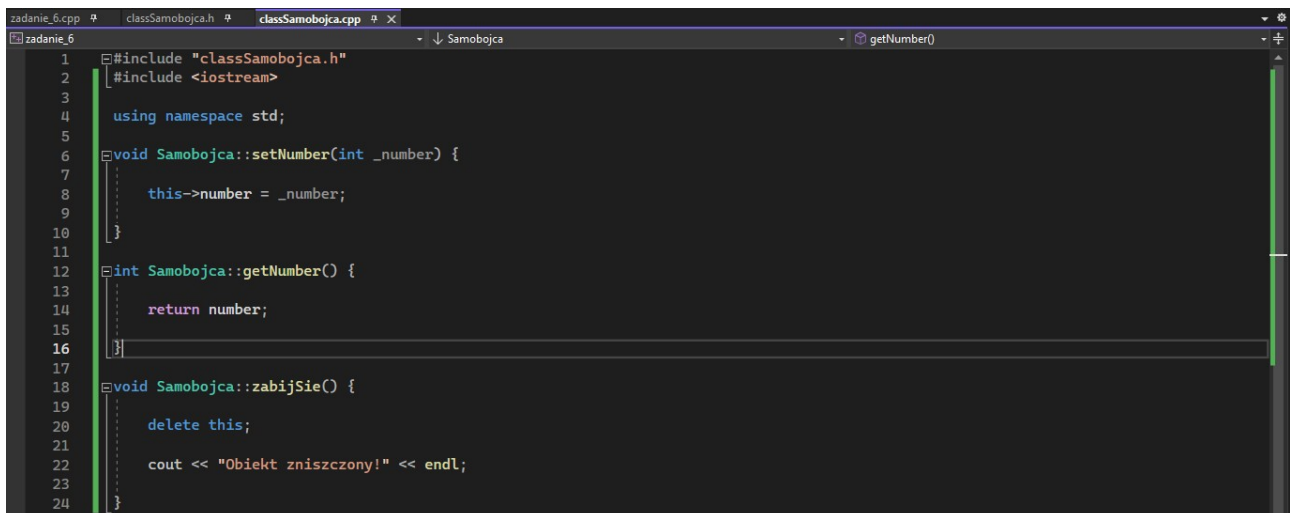
Utworzyć klasę o nazwie "Samobojca", w której metoda o nazwie "zabijSie" niszczy własny obiekt (musi być dynamiczny!). Omówić działanie i ewentualne problemy.

Rozwiązanie zadania 6

A screenshot of a code editor showing the file 'zadanie_6.cpp'. The code includes <iostream> and 'classSamobojca.h', uses the std namespace, and defines a main function. In main, a Samobojca object 'sdyn' is dynamically allocated, its number is set to 200, it is printed, then 'zabijSie()' is called, it is printed again, and finally deleted. The source path is shown as 'source: ...\Modul 3\Labaratorium 3\Zadanie 6\zadanie_6.cpp'.

```
1 #include <iostream>
2 #include "classSamobojca.h"
3
4 using namespace std;
5
6 int main() {
7     Samobojca * sdyn;
8
9     sdyn = new Samobojca;
10    sdyn->setNumber(200);
11    cout << "Numer sdyn: " << sdyn->getNumber() << endl;
12
13    sdyn->zabijSie();
14    cout << "Numer sdyn: " << sdyn->getNumber() << endl;
15
16    delete sdyn;
17
18    return 0;
19 }
```

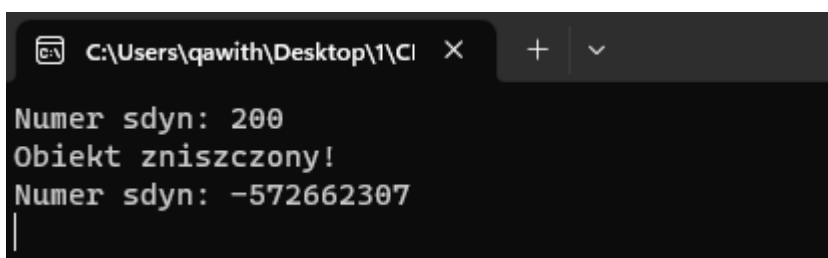
source: ...\Modul 3\Labaratorium 3\Zadanie 6\zadanie_6.cpp

A screenshot of a code editor showing the file 'classSamobojca.cpp'. It includes 'classSamobojca.h' and <iostream>, uses the std namespace, and defines three methods for the Samobojca class: setNumber, getNumber, and zabijSie. zabijSie calls delete on 'this' and prints a message. The source path is shown as 'source: ...\Modul 3\Labaratorium 3\Zadanie 6\classSamobojca.cpp'.

```
1 #include "classSamobojca.h"
2 #include <iostream>
3
4 using namespace std;
5
6 void Samobojca::setNumber(int _number) {
7     this->number = _number;
8 }
9
10
11
12 int Samobojca::getNumber() {
13     return number;
14 }
15
16
17
18 void Samobojca::zabijSie() {
19     delete this;
20
21     cout << "Obiekt zniszczony!" << endl;
22 }
23
24 }
```

source: ...\Modul 3\Labaratorium 3\Zadanie 6\classSamobojca.cpp

Wyjście:

A screenshot of a Windows command prompt window showing the output of the program. The output consists of three lines: 'Numer sdyn: 200', 'Obiekt zniszczony!', and 'Numer sdyn: -572662307'. The window title is 'C:\Users\qawith\Desktop\1\Cl' and it has standard window controls.

```
C:\Users\qawith\Desktop\1\Cl
Numer sdyn: 200
Obiekt zniszczony!
Numer sdyn: -572662307
```

Zadanie 7

Lista inicjalizacyjna konstruktora.

Zadanie najlepiej zapisać w jednym pliku.

W tym konstruktorze spróbować nadać stałej DANE wartość np. 11 za pomocą kodu umieszczonego w ciele konstruktora. Efekt? Powód?

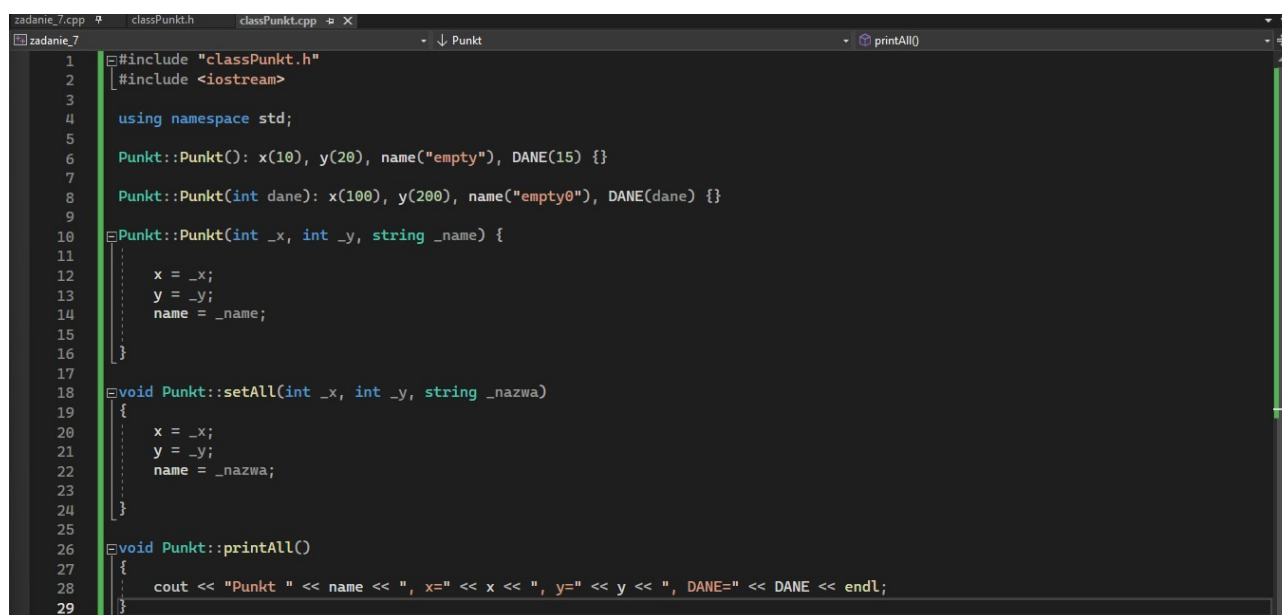
Kolejno spróbować ustawić stałą w tym samym konstruktorze ale w liście inicjalizacyjnej. Efekt? Powód?

Do zastosowania listy inicjalizacyjnej powrócimy podczas omawiania kwestii dziedziczenia.

Rozwiązanie zadania 7

W tym konstruktorze spróbować nadać stałej DANE wartość np. 11 za pomocą kodu umieszczonego w ciele konstruktora. Efekt? Powód?

Kolejno spróbować ustawić stałą w tym samym konstruktorze ale w liście inicjalizacyjnej. Efekt? Powód?



```
1 #include "classPunkt.h"
2 #include <iostream>
3
4 using namespace std;
5
6 Punkt::Punkt(): x(10), y(20), name("empty"), DANE(15) {}
7
8 Punkt::Punkt(int dane): x(100), y(200), name("empty0"), DANE(dane) {}
9
10 Punkt::Punkt(int _x, int _y, string _name) {
11     x = _x;
12     y = _y;
13     name = _name;
14 }
15
16
17
18 void Punkt::setAll(int _x, int _y, string _nazwa)
19 {
20     x = _x;
21     y = _y;
22     name = _nazwa;
23 }
24
25
26 void Punkt::printAll()
27 {
28     cout << "Punkt " << name << ", x=" << x << ", y=" << y << ", DANE=" << DANE << endl;
29 }
```

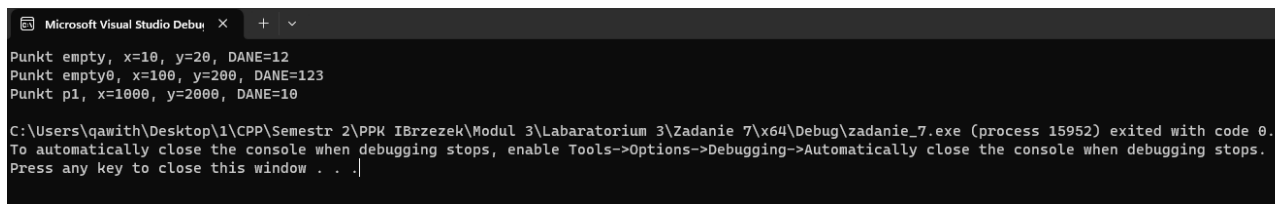
source: ...\\Modul 3\\Labaratorium 3\\Zadanie 7\\classPunkt.cpp



```
1 #include <iostream>
2 #include "classPunkt.h"
3
4 using namespace std;
5
6 int main() {
7     Punkt p1, p2(123), p3(1000, 2000, "p1");
8
9     p1.printAll();
10    p2.printAll();
11    p3.printAll();
12    return 0;
13 }
```

source: ...\\Modul 3\\Labaratorium 3\\Zadanie 7\\zadanie_7.cpp

Wyjście:



```
Microsoft Visual Studio Debu  X  +  v

Punkt empty, x=10, y=20, DANE=12
Punkt empty0, x=100, y=200, DANE=123
Punkt p1, x=1000, y=2000, DANE=10

C:\Users\qawith\Desktop\1\CPP\Semestr 2\PPK IBrzezek\Modul 3\Labaratorium 3\Zadanie 7\x64\Debug\zadanie_7.exe (process 15952) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .|
```

W ciele konstruktora klasy **Punkt**, nie można zmienić wartości stałej **DANE**, ponieważ stałe mają **CONST**. Natomiast w liście inicjalizacyjnej konstruktora można ustawić wartość stałej **DANE** dla każdego obiektu.