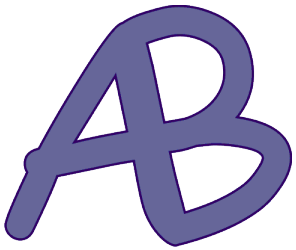
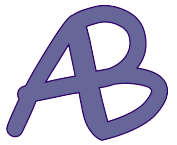
WYKŁAD



dr Artur Bartoszewski
Katedra Informatyki
UTH Radom

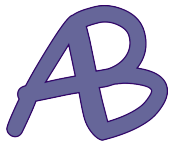
Przekazywanie do funkcji argumentów będących obiektami

Przekazywanie obiektów do funkcji



Obiekt jest w pewnym sensie rozbudowaną wersją zmiennej. Podobnie jak zmienne typu int czy double, każdy obiekt można przekazać do funkcji jako argument (mowa tu o funkcji z poza klasy, nie metodzie wchodzącej w skład klasy).

Przekazywanie obiektów do funkcji

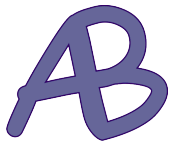


Przekazywanie obiektu przez wartość

Domyślnie przesyłamy obiekt przez wartość – czyli na potrzeby funkcji tworzona jest jego kopia (tak samo jak w przypadku zmienne prostej)

Uwaga: Jeśli obiekt jest duży, to proces kopiowania może trwać dłużej. Wielokrotne wysłanie przez wartość może wyraźnie wpływać na zwolnienie programu.

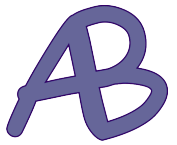
Przekazywanie obiektów do funkcji



```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  class osoba
6  {
7  private:
8      string nazwisko;
9      string imie;
10     int wiek;
11 public:
12     osoba(string im="NN", string nazw="NN", int w=0)
13        :imie(im),nazwisko(nazw),wiek(w) {}
14
15     string getImie() {return imie;}
16     string getNazwisko() {return nazwisko;}
17     int getwiek() {return wiek;}
18     void setImie(string imie) {this->imie=imie;}
19     void setNazwisko(string nazwisko) {this->nazwisko=nazwisko;}
20     void setWiek(int wiek) {this->wiek=wiek;}
21 };
```

Przekazywanie obiektu przez wartość - przykład

Przekazywanie obiektów do funkcji



```
23  void rodo(osoba temp);
24  int main()
25  {
26        osoba ktos("Jan", "Kowalski", 30);
27        rodo(ktos);
28        cout << ktos.getImie()<<" "
29             << ktos.getNazwisko()<<" "
30             << ktos.getwiek()<<endl;
31      return 0;
32  }
33  void rodo(osoba temp)
34  {
35      string s = temp.getNazwisko();
36      for (int i=1; i<s.length(); i++)
37          s[i]='x';
38      temp.setNazwisko(s);
39  }
```

Obiekt klasy „Osoba” przekazujemy do funkcji „rodo()” wewnątrz której jest modyfikowany

Niestety nie zadziałało….

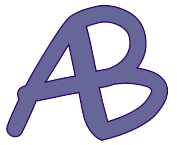
Przekazywanie obiektów do funkcji

```
32  }
33  void rodo(osoba temp)
34  {
35      string s = temp.getNazwisko();
36      for (int i=1; i<s.length(); i++)
37          s[i]='x';
38      temp.setNazwisko(s);
39  }
```

Należy pamiętać, że przekazując obiekt przez wartość wysyłamy do funkcji jego kopię, która jest usuwana z pamięci po zakończeniu funkcji.

Zmiany wprowadzone na kopii nie wpłyną na oryginał.

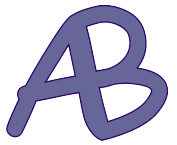
Przekazywanie obiektów do funkcji



Przekazywanie obiektu przez referencję

Referencja jest drugą nazwą, „przezwiskiem” - nie przezwiskiem klasy, ale danego egzemplarza jej obiektu. Wysyłając taki egzemplarz obiektu do funkcji na zasadzie przesłania przez referencję - sprawiamy, że nie jest on kopiowany. Funkcja ma dostęp do oryginału.

Przekazywanie obiektów do funkcji



```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  class osoba
6  {
7  private:
8      string nazwisko;
9      string imie;
10     int wiek;
11 public:
12     osoba(string im="NN", string nazw="NN", int w=0)
13        :imie(im),nazwisko(nazw),wiek(w) {}
14
15     string getImie() {return imie;}
16     string getNazwisko() {return nazwisko;}
17     int getwiek() {return wiek;}
18     void setImie(string imie) {this->imie=imie;}
19     void setNazwisko(string nazwisko) {this->nazwisko=nazwisko;}
20     void setWiek(int wiek) {this->wiek=wiek;}
21 };
```

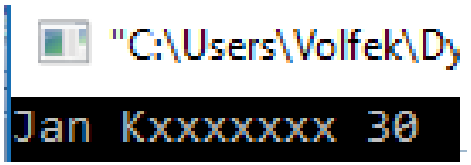
Przekazywanie obiektu przez referencję - przykład

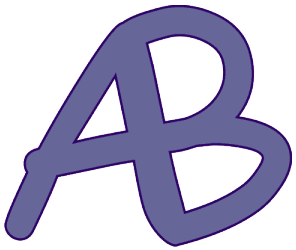
Klasa wygląda dokładnie tak samo jak w poprzednim przykładzie

Przekazywanie obiektów do funkcji

Przekazujemy obiekt przez referencję – tym razem zadziała prawidłowo

```
23  void rodo(osoba &ktos);
24  int main()
25  {
26      osoba ktos("Jan", "Kowalski", 30);
27      rodo(ktos);
28      cout << ktos.getImie()<<" "
29           << ktos.getNazwisko()<<" "
30           << ktos.getwiek()<<endl;
31      return 0;
32  }
33  void rodo(osoba &temp)
34  {
35      string s = temp.getNazwisko();
36      for (int i=1; i<s.length(); i++)
37          s[i]='x';
38      temp.setNazwisko(s);
39  }
```





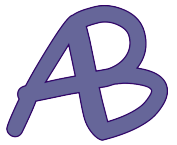
dr Artur Bartoszewski
Katedra Informatyki
UTH Radom

Przykład gra RPG v 0.1

Projekt będzie rozwijany ;)

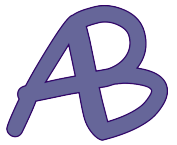
Tworzymy bardzo prostą tekstową grę RPG, w której będziemy wystawiać dwa obiekty klasy „postac” do walki w funkcji „ring”

Przekazywanie obiektów do funkcji - przykład



```
1   #include <iostream>
2   #include<cstring>
3   #include<ctime>
4   #include<stdlib.h>
5   using namespace std;
6
7   class postac
8   {
9       char imie[20];
10      int zycie;
11      int atak;
12      int obrona;
13    public:
14      postac(char *kto);
15      void setzycie(int ile) {zycie= ile;}
16      int getzycie() {return zycie;}
17      int getatak() {return atak;}
18      int getobrona() {return obrona;}
19      void przedstaw_sie ();
20      ~postac();
21  };
22
```

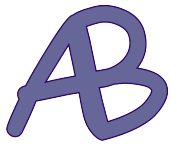
Przekazywanie obiektów do funkcji - przykład



```
23  postac::postac(char *kto)
24  {
25      strcpy(imie, kto);
26      zycie = (rand() % 30)+20;
27      atak = (rand() % 10)+5;
28      obrona = (rand()%5);
29  }
30
31  void postac::przedstaw_sie ()
32  {
33      cout <<endl<<imie<<" : "<<zycie<<" : "<<atak
34           <<" : "<<obrona;
35
36  }
37
38  postac::~postac()
39  {
40      cout <<endl<<"gracz: ";
41      przedstaw_sie();
42      cout <<": schodzi z ringu" <<endl;
43  }
```

Losujemy parametry postaci

Przekazywanie obiektów do funkcji - przykład



```
45  void ring(postac zawodnik1, postac zawodnik2);
46  int main()
47  {
48      srand(time(NULL));
49      postac gracz1("Conan the Librarian");
50      postac gracz2("Chuck z Texasu");
51      gracz1.przedstaw_sie();
52      gracz2.przedstaw_sie();
53      ring(gracz1,gracz2);
54      return 0;
55  }
```

```
57  void ring(postac zawodnik1, postac zawodnik2)
58  {
59      int z1,z2;
60      while (zawodnik1.getzycie()>=0 && zawodnik2.getzycie()>=0)
61      {
62          zawodnik1.setzycie(zawodnik1.getzycie()-(zawodnik2.getatak()-zawodnik1.getobrona()));
63          zawodnik2.setzycie(zawodnik2.getzycie()-(zawodnik1.getatak()-zawodnik2.getobrona()));
64      }
65  }
```

Przekazywanie obiektów do funkcji - przykład

```
45    void ring(postac zawodnik1, postac zawodnik2);
```

Ponieważ obiekty przesłane zostały przez wartość widzimy cztery wywołania destruktora :

- dwa dla kopii użytych w funkcji ring()
- dwa dla oryginałów.

```
Conan the Librarian : 20 : 5 : 1
Chuck z Texasu : 36 : 14 : 0
gracz:
Conan the Librarian : -6 : 5 : 1: schodzi z ringu

gracz:
Chuck z Texasu : 26 : 14 : 0: schodzi z ringu

gracz:
Chuck z Texasu : 36 : 14 : 0: schodzi z ringu

gracz:
Conan the Librarian : 20 : 5 : 1: schodzi z ringu
```

Przekazywanie obiektów do funkcji - przykład

```
45  void ring(postac &zawodnik1, postac &zawodnik2);
46  int main()
47  {
48      srand(time(NULL));
49      postac gracz1("Conan the Librarian");
50      postac gracz2("Chuck z Texasu");
51      gracz1.przedstaw_sie();
52      gracz2.przedstaw_sie();
53      ring(gracz1,gracz2);
54      return 0;
55  }
```

```
57  void ring(postac &zawodnik1, postac &zawodnik2)
58  {
59      int z1,z2;
60      while (zawodnik1.getzycie()>=0 && zawodnik2.getzycie()>=0)
61      {
62          zawodnik1.setzycie(zawodnik1.getzycie()-(zawodnik2.getatak()-zawodnik1.getobrona()));
63          zawodnik2.setzycie(zawodnik2.getzycie()-(zawodnik1.getatak()-zawodnik2.getobrona()));
64      }
65  }
```

Przekazywanie obiektów do funkcji - przykład

```
45    void ring(postac &zawodnik1, postac &zawodnik2);
```

Ponieważ obiekty przesłane zostały przez referencję widzimy tylko dwa wywołania destruktora - dwa dla oryginałów w chwili zamykania programu

```
Conan the Librarian : 33 : 8 : 1
Chuck z Texasu : 35 : 9 : 3
gracz:
Chuck z Texasu : 10 : 9 : 3: schodzi z ringu

gracz:
Conan the Librarian : -7 : 8 : 1: schodzi z ringu
```

Literatura:

W prezentacji wykorzystano przykłady i fragmenty:

- Gręboszj J. : ***Symfonia C++, Programowanie w języku C++ orientowane obiektowo***, Wydawnictwo Edition 2000.
- Jakubczyk K.: *Turbo Pascal i Borland C++ Przykłady*, Helion.

Warto zajrzeć także do:

- Sokół R. : ***Microsoft Visual Studio 2012 Programowanie w Ci C++,*** Helion.
- Kerninghan B. W., Ritchie D. M.: ***język ANSI C***, Wydawnictwo Naukowo Techniczne.

Dla bardziej zaawansowanych:

- Gręboszj J. : ***Pasja C++,*** Wydawnictwo Edition 2000.
- Meyers S.: ***język C++ bardziej efektywnie***, Wydawnictwo Naukowo Techniczne