Young Programmer

www.inceptio.org.pl

Organizator:

fundacja INCEPTIO

Partnerzy:

Wydział Fizyki Politechnika Warszawska

Stowarzyszenie KLATRAT

Dofinansowanie:

Miasto Stołeczne Warszawa

Zakochaj się w Warszawie

Young Programmer

www.inceptio.org.pl

Zajęcia nr 2

Programowanie strukturalne

dr inż. Łukasz Graczykowski
mgr inż. Leszek Kosarzewski
Wydział Fizyki Politechniki Warszawskiej

Organizator:

Partnerzy:

Dofinansowanie:

MIASTO STOŁECZNE WARSZAWA

Pętla while

```
#include <iostream>
using namespace std;

int main ()
{
  bool koniec = false;

  while(!koniec){
    cout << "Co chcesz zrobić?" << endl;
    cout << "1 - dodawanie" << endl;
    cout << "2 - wyjść" << endl;
    char a;
    cin >> a;
    switch(a){
    case '1':
      cout<<"Podaj 1 liczbe:"<<endl;
      double x;
      cin>>x;
      cout<<"Podaj 2 liczbe:"<<endl;
      double y;
      cin>>y;
      cout<<"Podaj wynik: "<<x<<" + "<<y<<" = ";
      double z;
      cin>>z;
      if(z==x+y) cout<<endl<<"Dobrze!"<<endl;
      else cout<<"Pomylka! Prawidlowy wynik to: "<<x+y<<endl;
      break;

    case '2': koniec = true;
      break;
    default: cout<<"Wpisz 0 lub 1!"<<endl;
    }
  }

  return 0;
}
```

Wyrażenie kontrolne

początek pętli

Pętla while wykonuje się dopóki wyrażenie kontrolne jest prawdziwe.

zmiana wartości wyrażenia kontrolnego

koniec pętli

Pętla while

```
#include <iostream>
using namespace std;

int main ()
{
  bool koniec = false;

  while(!koniec){
    cout << "Co chcesz zrobić?" << endl;
    cout << "1 - dodawanie" << endl;
    cout << "2 - wyjść" << endl;
    char a;
    cin >> a;
    switch(a){
    case '1':
      cout<<"Podaj 1 liczbe:"<<endl;
      double x;
      cin>>x;
      cout<<"Podaj 2 liczbe:"<<endl;
      double y;
      cin>>y;
      cout<<"Podaj wynik: "<<x<<" + "<<y<<" = ";
      double z;
      cin>>z;
      if(z==x+y) cout<<endl<<"Dobrze!"<<endl;
      else cout<<"Pomylka! Prawidlowy wynik to: "<<x+y<<endl;
      break;

    case '2': koniec = true;
      break;
    default: cout<<"Wpisz 0 lub 1!"<<endl;
    }
  }

  return 0;
}
```

Pętla while wykonuje się
dopóki wyrażenie kontrolne jest prawdziwe.

koniec = false
zatem
!koniec = true

Pętla wykonuje się

koniec = true
zatem
!koniec = false

Pętla nie wykonuje się

Pętla while

```
#include <iostream>
using namespace std;

int main ()
{
  bool koniec = false;

  while(!koniec){
    cout << "Co chcesz zrobić?" << endl;
    cout << "1 - dodawanie" << endl;
    cout << "2 - wyjść" << endl;
    char a;
    cin >> a;
    switch(a){
    case '1':
      cout<<"Podaj 1 liczbe:"<<endl;
      double x;
      cin>>x;
      cout<<"Podaj 2 liczbe:"<<endl;
      double y;
      cin>>y;
      cout<<"Podaj wynik: "<<x<<" + "<<y<<" = ";
      double z;
      cin>>z;
      if(z==x+y) cout<<endl<<"Dobrze!"<<endl;
      else cout<<"Pomylka! Prawidlowy wynik to: "<<x+y<<endl;
      break;

    case '2': koniec = true;
      break;
    default: cout<<"Wpisz 0 lub 1!"<<endl;
    }
  }

  return 0;
}
```

Gdyby: **koniec = true**
albo
Gdyby: **while(koniec)**

Pętla while wykonuje się
dopóki wyrażenie kontrolne jest prawdziwe.

Jeśli wyrażenie kontrolne od początku będzie Fałszywe, to pętla while się nie wykona!

Pętla do-while

```
#include <iostream>
using namespace std;

int main ()
{
  bool koniec = false;

  do{
    cout << "Co chcesz zrobić?" << endl;
    cout << "1 - dodawanie" << endl;
    cout << "2 - wyjść" << endl;
    char a;
    cin >> a;
    switch(a){
    case '1':
      cout<<"Podaj 1 liczbe:"<<endl;
      double x;
      cin>>x;
      cout<<"Podaj 2 liczbe:"<<endl;
      double y;
      cin>>y;
      cout<<"Podaj wynik: "<<x<<" + "<<y<<" = ";
      double z;
      cin>>z;
      if(z==x+y) cout<<endl<<"Dobrze!"<<endl;
      else cout<<"Pomylka! Prawidlowy wynik to: "<<x+y<<endl;
      break;

    case '2': koniec = true;
      break;
    default: cout<<"Wpisz 0 lub 1!"<<endl;
    }while(!koniec);
  }

  return 0;
}
```

Po uruchomieniu następuje **wykonanie** instrukcji

Pętla do-while
zawsze wykonuje się **za pierwszym razem**
oraz dopóki wyrażenie kontrolne jest prawdziwe.

Sprawdzenie warunku.
Jeśli warunek **!koniec = true**,
to pętla **wykona się** ponownie.

Funkcje

```
#include <iostream>
using namespace std;

int main()
{
    int dzien, miesiac, rok;

    cout<<"Podaj date swoich urodzin."<<endl;
    cout<<"Podaj dzien:"<<endl;
    cin>>dzien;
    cout<<"Podaj miesiac:"<<endl;
    cin>>miesiac;
    cout<<"Podaj rok:"<<endl;
    cin>>rok;

    cout<<"Urodziles sie
"<<dzien<<"."<<miesiac<<"."<<rok<<" roku."<<endl;

    return 0;
}
```

Funkcja główna każdego programu.

Po wykonaniu funkcji main(),
nie zwraca ona żadnych wartości (return 0;)

Funkcje

```
#include <iostream>
using namespace std;

int dane()
{
    int zmienna;
    cin>>zmienna;
    return zmienna;
}

int main()
{
    int dzien, miesiac, rok;

    cout<<"Podaj date swoich urodzin."<<endl;
    cout<<"Podaj dzien:"<<endl;
    dzien = dane();
    cout<<"Podaj miesiac:"<<endl;
    miesiac = dane();
    cout<<"Podaj rok:"<<endl;
    rok = dane();

    cout<<"Urodziles sie
"<<dzien<<"."<<miesiac<<"."<<rok<<" roku."<<endl;

    return 0;
}
```

Funkcja zewnętrzna typu **int**.
Typ mówi o tym, jakiego typu dane będą zwracane przez funkcję.

Po wykonaniu funkcji dane(), zwraca ona wartości obiektu **zmienna**.

Wywołujemy funkcję dane(). Wartość, którą ona zwraca, przypisujemy obiektowi **dzien**.

Funkcje

```
#include <iostream>
using namespace std;

int dane(string pytanie)
{
    int zmienna;
    cout<<pytanie<<endl;
    cin>>zmienna;
    return zmienna;
}

int main()
{
    int dzien, miesiac, rok;

    cout<<"Podaj date swoich urodzin."<<endl;
    dzien = dane("Podaj dzien:");
    miesiac = dane("Podaj miesiac:");
    rok = dane("Podaj rok:");

    cout<<"Urodziles sie
"<<dzien<<"."<<miesiac<<"."<<rok<<" roku."<<endl;

    return 0;
}
```

Funkcja zewnętrzna typu int.
Do wywołania potrzebuje zmiennej typu string.

Obiekt pytanie typu string został zdefiniowany tylko na potrzeby funkcji zewnętrznej.

Wywołujemy funkcję dane(string), która oczekuje wartości typu string. Funkcja zwraca wartość obiektu zmienna.

Funkcje

Funkcję, jak zmienną można najpierw zadeklarować.

Wtedy podajemy jej typ (np. **int**), jej nazwę (np. dane), oraz w nawiasie typ zmiennej którą funkcja oczekuje (np. **string**).

```
int dane(string);

int dane(string pytanie)
{
    int zmienna;
    cout<<pytanie<<endl;
    cin>>zmienna;
    return zmienna;
}

int main() { … }
```

Definicja funkcji.

Z deklaracji funkji korzystamy zazwyczaj, gdy program jest bardzo rozwinięty, oraz gdzie podzielimy go na kilka plików.

Referencje

Referencja obiektu danego typu to takie **przezwisko obiektu**.

Wykonanie operacji na referencji daje taki sam efekt jak działanie na zmiennej do której odnosi się referencja.

```
int main(){
int zmienna = 7;
inr &ref = zmienna;

cout<<"zmienna = "<<zmienna<<", a referencja = "<<ref<<endl;

ref = ref - 2;

cout<<"zmienna = "<<zmienna<<", a referencja = "<<ref<<endl;
return 0;
}
```

Referencję oznaczamy znaczkiem &
ref – nazwa referencji
zmienna – obiekt do którego odnosi się referencja ref

Referencję wywołujemy jak zwykłą zmienną.

Wynik działania programu:

zmienna = 7, a referencja = 7
zmienna = 5, a referencja = 5

Referencje w funkcjach

Referencję w funkcji wykorzystujemy następująco:

Deklaracja funkcji:

```
void funkcja(int &);
```

Tu nie podajemy nazwy zmiennej. Podajemy tylko operator referencji &.

Definicja funkcji:

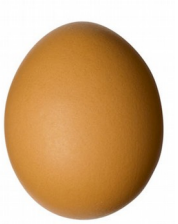
```
void funkcja(int &ref)
{
    cout<<ref<<endl;
}
```

Tu podajemy operator referencji i jej nazwę.

W ciele funkcji korzystamy już tylko z nazwy referencji.

Tablice

Tworzymy 1 obiekt, np. jajko



```
int jajko;
```

Jeśli chcemy mieć mendel jajek, to trzeba by stworzyć wiele obiektów:

```
int jajko1, jajko2, jajko3, jajko4, jajko5, jajko6,
jajko7, jajko8, jajko9, jajko10, jajko11, jajko12,
jajko13, jajko14, jajko15;
```

Niepraktyczny chaos i dużo pisania

Stwórzmy więc tablicę 15 elementów typu int:

```
int jajko[10];
```

Typ obiektu

Nazwa obiektu

Ilość elementów w tablicy

Operator tablicy obiektów, oznacza się go [ ]

Tablice

Nadawanie wartości elementom tablicy.

```
int liczba[4] = {12, 23, 32, 7};
```

Wartości elementów tablicy:

liczba[0] ***ma wartość*** **12**
liczba[1] ***ma wartość*** **23**
liczba[2] ***ma wartość*** **32**
liczba[3] ***ma wartość*** **7**

Elementy tablicy zaczynają się od **0**!
Ostatni element to n-1, gdzie tablica[n].

Jeśli napiszemy:

```
int liczba[4] = {12,23};
```

liczba[0] *ma wartość* 12
liczba[1] *ma wartość* 23
pozostałe mają wartość 0

Nadawanie wartości elementom tablicy.

```
int liczba[4] = {12, 23, 32, 7};
```

Jednoczesna definicja i inicjalizacja tablicy

Nie podaliśmy wymiaru tablicy.
Kompilator sam policzy ilość elementów i nada odpowiedni rozmiar tablicy (w tym przypadku 4)

Przekazanie tablicy do funkcji

Mamy funkcję, która spodziewa się argumentu typu double w postaci tablicy.

```
void funkcja( double tt[] );
```

Wywołanie takiej funkcji:

```
double tablica[] = {1, 2, 3, 4};
funkcja(tablica);
```

Definicja i inicjalizacja tablicy

Wywołanie funkcji

Korzystamy tylko z nazwy tablicy.
Nazwa tablicy jest jednocześnie **adresem jej zerowego elementu**!

Element tablicy o indeksie 2: tablica[2]
Adres elementu tablicy o indeksie 2: &tablica[2]

& - oznacza uzyskanie adresu obiektu

Young Programmer

www.inceptio.org.pl

Zajęcia nr 2

Podstawy programowania

Zadania

Organizator:

fundacja INCEPTIO

Partnerzy:

Wydział Fizyki Politechniki Warszawskiej

Stowarzyszenie KLATRAT

Dofinansowanie:

Miasto Stołeczne Warszawa

Zakochaj się w Warszawie

Zadanie 1. Ile lat jesteś starszy od kolegi?

Napisz program, który sprawdza o ile jesteś starszy od kolegi.

Należy napisać funkcję zewnętrzną typu int, która pobiera obiekt typu string (np. **int dane(string pytanie)**). Funkcja pobiera pytanie, które następnie wypisuje za pomocą komendy cout. Pobiera ona zmienną typu int, którą użytkownik wprowadzi z klawiatury (rok urodzenia). Funkcja ma zwracać pobraną zmienną.

W głównej funkcji int main() prosimy użytkownika o podanie roku swoich urodzin za pomocą zewnętrznej funkcji **dane(string)**. Wartość zwracanej zmiennej przypisujemy do zmiennej **int rok_ja**. Następnie w taki sam sposób pobieramy rok urodzenia kolegi i przypisujemy tę wartość do zmiennej **int rok_ja**.
Liczymy różnicę lat i przypisujemy ją do nowej zmiennej.

Korzystamy z warunku **if-else**, aby wyświetlić (za pomocą cout) odpowiednią odpowiedź dla użytkownika, zawierającą obliczoną różnicę lat i wnioski. Jedna odpowiedź wypisana powinna wskazywać, że użytkownik jest starszy od kolegi, następna, że jest młodszy, a ostatnia, że urodzili się w tym samym roku np.:

```
if(warunek1){ ... }
else if(warunek2){ … }
else { ... }
```

Zadanie 2. Przesunięcie punktu.

Napisz program, który pobiera współrzędną punktu na osi OX, a następnie przesuwa ją o zadaną wartość.

Napisz funkcję zewnętrzną, która pobiera zmienną typu string - pytanie do użytkownika (np. **double pobierz(string pytanie)**). Zwraca zmienną typu double pobraną z klawiatury.

Napisz funkcję zewnętrzną, która pobiera zmienną typu double (np. **void wypisz(double a)**). Wypisuje ona na ekran (za pomocą komendy cout) informację o położeniu punktu na osi.

Napisz funkcję zewnętrzną pobierającą dwie zmienne typu double (położenie punktu na osi **a** i wartość przesunięcia **b**) np. **void przesuniecie(double a, double b)**. Funkcja wypisuje na ekran współrzędną punktu przed przesunięciem. Następnie realizuje matematyczną operację odpowiadającą przesunięciu punktu. Ostatecznie wypisuje na ekran wartość po przesunięciu.

Napisz kolejną funkcję zewnętrzną pobierającą referencję do zmiennej typu double oraz zmienną typu double (referencję do położenie punktu na osi **a** i wartość przesunięcia **b**) np. **void przesuniecie2(double &a, double b)**. Funkcja wypisuje na ekran współrzędną punktu przed przesunięciem. Następnie realizuje matematyczną operację odpowiadającą przesunięcie punktu. Ostatecznie wypisuje na ekran wartość po przesunięciu.

W funkcji głównej int main() pobieramy współrzędną położenia punktu na osi OX za pomocą funkcji **pobierz** i przypisujemy ją do zmiennej typu double np. **wsp_x**. Informujemy użytkownika (za pomocą komendy cout) o pierwszej operacji przesunięcia punktu o **wsp_x**, o wartość **wsp_x2 = 5.0** (albo inną wartość zmiennoprzecinkową). Realizujemy operację przesunięcia na punkcie funkcją **przesuniecie(double, double)**. Wypisujemy wartość współrzędnej za pomocą funkcji **wypisz(double)**. Następnie postępujemy tak samo, ale przesuwamy punkt za pomocą drugiej funkcji – **przesuniecie2(double &, double)**.

Zadanie 3.a. Rzut kostką

Napisz program realizujący rzut kostką n-ścienną.

a)Wylosuj liczbę oczek.
Skorzystaj z instrukcji preprocesora umożliwiającej generację liczb pseudolosowych (**#include <stdlib.h>**).

W funkcji głównej zdeklaruj dwie zmienne typu **int sciany, rzut;** , które będą przechowywać informacje o ilości ścian kostki i wyniku rzutu kością.

Wywołaj funkcję **srand(time(0));** Ustawia ona punkt początkowy generatora liczb pseudolosowych na wartości 0. Co więcej każda generacja będzie niepowtarzalna (na miarę generatora liczb pseudolosowych).

Poproś użytkownika o podanie ilości ścian kostki i przypisz ją do zmiennej **sciany** (za pomocą komend cout i cin).

Skorzystaj z pętli do-while z warunkiem (koniec =='t'). W instrukcji pętli wylosuj liczbę oczek w kostce za pomocą komendy **rand()% sciany + 1;** . Wylosowaną wartość wypisz na ekran. Zapytaj się użytkownika, czy chce kontynuować (niech wybierze odpowiedz t/n) i pobierz odpowiedź.

Zadanie 3.b. Rzut kostką

b)Wylosuj liczbę oczek. Oblicz sumę z wszystkich losowań i wypisz ich liczbę.

Przerób poprzedni program. Instrukcję losowania przenieś do funkcji zewnętrznej, która pobiera wartość int (ilość oczek) oraz zwraca wynik losowania w formie zmiennej typu int (**return wartosc;**).

W funkcji main() w instrukcji pętli dopisz kilka linii. Po rzucie kostką dodaj wynik rzutu do nowej zmiennej **int suma**. Następnie zwiększ wartość nowej zmiennej **int n** o 1 (podczas definiowania zmiennej **n** nadaj jej wartość 0).

Przed zakończeniem głównej funkcji main() wypisz na ekran informację o liczbie wykonanych rzutów, liczbie ścian kostki oraz sumę rzutów.

Zadanie 3.c. Rzut kostką

c)Wylosuj liczbę oczek. Oblicz sumę z wszystkich losowań i wypisz ich liczbę. Zapamiętaj informację o wynikach w tablicy.

Przerób poprzedni program. W głównej funkcji main() stwórz jednowymiarową tablicę **rzut[]** typu int o dużym rozmiarze np. 100. Pamiętaj o nadaniu początkowych wartości wszystkim elementom 0.

Wyniki losowań wpisuj do kolejnych elementów tablicy (skorzystaj ze zmiennej **n**).

Na końcu, przed instrukcją **return 0;** wypisz wszystkie zapisane wyniki na ekran. Skorzystaj z pętli **for**, nowego iteratora **int i** oraz zmiennej **n**.