Wojskowa
Akademia
Techniczna

Podstawy programowania

c.d. Analiza algorytmów.
Projektowanie algorytmów.

Wojskowa Akademia Techniczna

Algorytmy iteracyjne

Zad.1. Napisz program obliczający *n*-tą potęgę zadanej liczby *x*.

Zad.2. Napisz program, który będzie prosił o hasło dostępu do systemu dopóki nie otrzyma hasła poprawnego.

- gdy je otrzyma wyświetli napis „Zalogowano!”.
- w przypadku więcej niż 5 prób wypisze komunikat: „Dostęp zablokowany!”

Wojskowa Akademia Techniczna

Liczby losowe w C++

- Liczby „losowe” generowane przez komputer wyznaczane są na podstawie przyjętego algorytmu (wzoru generacyjnego), więc są przewidywalne. Z tego powodu nazywane są liczbami **pseudolosowymi**.
- W C++ za generację takich liczb odpowiada m.in. funkcja ***rand*()**, która zwraca liczbę całkowitą z przedziału od 0 do RAND_MAX (wykorzystywany jest równomierny rozkład prawdopodobieństwa).
- W celu zapewnienia zmienności generowanych liczb pomiędzy uruchomieniami programu konieczna jest inicjalizacja generatora liczbą o charakterze zmiennym, do czego dobrze nadaje się aktualny czas (funkcja ***time*()**). Za inicjalizację odpowiada funkcja ***srand*()**.
- deklaracje funkcji ***srand*()** i ***rand*()** znajdują się w pliku nagłówkowym cstdlib

```
printf ("First number: %d\n", rand()%100);
srand (time(NULL));
printf ("Random number: %d\n", rand()%100);
srand (1);
printf ("Again the first number: %d\n", rand()%100);
```

Possible output:

```
First number: 41
Random number: 13
Again the first number: 41
```

Zad.3. Napisz program, który wygeneruje trzy liczby losowe: pierwszą z przedziału od 0 do 99, drugą z przedziału od 1 do 100, trzecią z przedziału od 1985 do 2014.

Wojskowa Akademia Techniczna

Podstawy programowania

Widoczność, modyfikatory i klasy zmiennych.

Wskaźniki i zastosowanie tablic statycznych.

Materiał zaczerpnięty z http://edu.pjwstk.edu.pl/wyklady

Wojskowa Akademia Techniczna

Zasięg i widoczność zmiennych

- **Zasięg** (ang. *scope*) deklaracji to ta część programu, gdzie deklaracja jest aktywna, czyli nazwa deklarowana może być użyta i odnosi się do tej właśnie deklaracji.
- Zakres **widoczności** (ang. *visibility*) natomiast to ten fragment, w których nazwa może być użyta bez żadnej kwalifikacji. Zakres widoczności zadeklarowanej zmiennej może być mniejszy niż zasięg deklaracji na skutek przesłonięcia.

Wojskowa Akademia Techniczna

Zasięg i widoczność zmiennych

- Zmienne lokalne są deklarowane wewnątrz funkcji, a ogólniej bloku ograniczonego nawiasami klamrowymi. Ich zasięg obejmuje część programu od miejsca deklaracji do końca tego bloku.
- Czasem życia *niestatycznej* zmiennej lokalnej jest czas od napotkania jej definicji do wyjścia z funkcji. W szczególności oznacza to, że kiedy przepływ sterowania powraca do tej samej funkcji (bloku), zmienne lokalne są tworzone na nowo i nie „pamiętają” swoich wcześniejszych wartości.
- Takimi zmiennymi są też zmienne deklarowane w części inicjalizacyjnej pętli **for,** jak i zmienne deklarowane w części warunkowej instrukcji **if**, **while**, **for**, **switch.**

Wojskowa Akademia Techniczna

Zasięg i widoczność zmiennych

- Zmienne globalne mogą być przesłonięte. Wówczas nazwa tej zmiennej w ciele funkcji odnosi się do zmiennej *lokalnej*. Zmienna globalna istnieje, ale jest w zakresie funkcji (bloku) niewidoczna.
- Do przesłoniętej zmiennej globalnej możemy odwołać się poprzez operator zasięgu ' ::' („czterokropek”). Jeśli na przykład przesłonięta została zmienna o nazwie ***k***, to do globalnej zmiennej o tej nazwie odwołać się można poprzez nazwę kwalifikowaną ***::k***.

Zasięg i widoczność zmiennych

```
  k: 0
::k: 0
  k: 10
::k: 0
  k: 10
::k: 1
```

```
W bloku:
  k: 77
::k: 1
Po bloku:
  k: 10
::k: 1
```

```
int k;

int main() {
    cout << "  k: " <<   k << endl;
    cout << "::k: " << ::k << endl;

    int k = 10;

    cout << "  k: " <<   k << endl;
    cout << "::k: " << ::k << endl;

    ::k = 1;

    cout << "  k: " <<   k << endl;
    cout << "::k: " << ::k << endl;
}
```

```
{
    int k = 77;
    cout << "W bloku:"      << endl;
    cout << "  k: " <<   k << endl;
    cout << "::k: " << ::k << endl;
}

cout << "Po bloku:" << endl;
cout << "  k: " <<   k << endl;
cout << "::k: " << ::k << endl;
```

Klasy pamięci i modyfikatory typow

- Klasa pamięci zmiennej (ang. *storage class specifier*) określana jest w deklaracji za pomocą specyfikatora, który jest jednym ze słów kluczowych **extern**,**register**, **static**.
- Zmienne zewnętrzne deklarowane są ze specyfikatorem **extern**. Stanowi ona informację dla kompilatora, że zmienna ta jest lub będzie *zdefiniowana* w innym pliku/module. Definicja z kolei może pojawić się tylko raz, jako definicja zmiennej globalnej *bez* specyfikatora **extern**.

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

double x1 = 11;
extern double x2;
void func();

int main()
{
    cout << "main: x1 = " << x1 << endl;
    cout << "main: x2 = " << x2 << endl;
    func();
}
```

PLIK 1

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

extern double x1;
double x2 = 22;

void func()
{
    cout << "func: x1 = " << x1 << endl;
    cout << "func: x2 = " << x2 << endl;
}
```

PLIK 2

Wojskowa Akademia Techniczna

Klasy pamięci i modyfikatory typow

- Zmienne statyczne deklarowane są ze specyfikatorem **static**. Mogą to być zarówno zmienne globalne, jak i lokalne.
- Zmienna statyczna jest tworzona tylko raz i inicjowana po utworzeniu zerem odpowiedniego typu. Jeśli jest to zmienna lokalna, to tworzona jest, gdy przepływ sterowania przechodzi po raz pierwszy przez jej deklarację. Jej czas życia to okres od utworzenia do końca programu.
- Zmienna statyczna lokalna, *nie* jest usuwana po wyjściu sterowania z tej funkcji.

```
int licznik;

void fun1() {
    static int licznik;
    licznik++;   // lokalna
    ::licznik++; // globalna
    cout << "Wywolan   fun1: " << licznik << endl;
}

void fun2() {
    static int licznik;
    licznik++;   // lokalna
    ::licznik++; // globalna
    cout << "Wywolan   fun2: " << licznik << endl;
}

int main() {
    fun1(); fun1(); fun2(); fun1(); fun2();
    cout << "Wywolan fun1/2: " << licznik << endl;
}
```

```
Wywolan   fun1: 1
Wywolan   fun1: 2
Wywolan   fun2: 1
Wywolan   fun1: 3
Wywolan   fun2: 2
Wywolan fun1/2: 5
```

Wojskowa Akademia Techniczna

Klasy pamięci i modyfikatory typow

- **volatile** i **const** są tzw. modyfikatorami typu (ang. *type qualifier*): nie wpływają na sposób przydziału pamięci czy linkowania, ale zmieniają typ deklarowanej zmiennej (choć reprezentacja bitowa pozostaje taka sama).
- Zmienne ulotne deklarowane są z modyfikatorem **volatile**. Modyfikator **volatile** oznacza, że zmienna ta może ulec zmianie „bez wiedzy” programu.

```
volatile double x;
```

Wojskowa Akademia Techniczna

Klasy pamięci i modyfikatory typow

- Zmienną każdego typu, zarówno „zwykłego”, jak i wskaźnikowego, można ustalić deklarując ją z modyfikatorem typu **const** przed lub za nazwą typu.
- **Wartości stałej (zmiennej ustalonej) nie można zmienić po jej utworzeniu.**
- Jedynym sposobem na nadanie stałej jej wartości jest inicjalizacja podczas jej definiowania, **bezpośrednio w instrukcji deklaracyjnej**.

```
const double PI = 3.1415926536; // TAK

const double PI;
PI = 3.1415926536; // NIE


const double x1 = 2*11;
const double x2 = x1 / 5.0;
const double x3 = sin(x2);
```

Wojskowa Akademia Techniczna

Wskaźniki

- Wartością zmiennej wskaźnikowej jest adres innej zmiennej.
- Deklaracja wskaźnika ma postać:

```
int *wskaznik;      char *px;      char **ppx;      MojaKlasa* wskaz;
```

- Typem wskaźnika px zadeklarowanego jako ' **Typ*** px;' jest '**Typ***'.
- Definicja wskaźnika ma postać

```
int liczba = 3;
int *wskaznik = &liczba;
```

- Znak ' &' pełni rolę **operatora wyłuskania adresu**
- Jeśli var jest identyfikatorem zmiennej, to wartością wyrażenia &var jest *adres* tej zmiennej.

Wojskowa Akademia Techniczna

Wskaźniki

- Zmienną, której adres jest wartością zmiennej wskaźnikowej, nazywamy zmienną wskazywaną przez ten wskaźnik.
- Znak ' *' pełni również rolę **operatora wyłuskania wartości.**

Zad. 1. Napisz program, w którym:
- zadeklarujesz zmienną typu całkowitego,
- zadeklarujesz zmienną wskaźnikową do zmiennej typu całkowitego,
- zmiennej wskaźnikowej przypiszesz adres zmiennej całkowitej,
- wypiszesz:
  - wartość zmiennej całkowitej
  - wartość zmiennej wskaźnikowej
  - wartość zmiennej całkowitej przy pomocy operatora wyłuskiwania wartości
  - adres zmiennej całkowitej przy pomoy operatora wyłuskania adresu
  - adres zmiennej wskaźnikowej

Wojskowa Akademia Techniczna

Wskaźniki

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

int main() {
    int x;
    int *px;
    px = &x;

    cout << "1. x = "  << x << endl;
    cout << "2. px = "  << px << endl;
    cout << "3. *px = "  << *px << endl;
    cout << "4. &x = "  << &x << endl;
    cout << "2. &px = "  << &px << endl;

    return 0;
}
```

Wojskowa Akademia Techniczna

Wskaźniki, a zmienne ustalone

- Zmienna zadeklarowana jako stała ma inny typ niż zmienna nieustalona, więc…

```
const int i = 25;
int *pi = &i; // NIE
```

- Przypisanie odwrotne jednak *jest* legalne. Można do wskaźnika do ustalonej zmiennej przypisać adres zmiennej nieustalonej. Zapobiega to przypadkowym zmianom wartości zmiennej wewnątrz funkcji.

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

  int fun(const int * pi) {
      //*pi = 2 * (*pi); // NIE !!!
      return *pi;
  }

  int main() {
      int i = 2;
      int res = fun(&i);
      cout << "res = " << res << endl;
  }
```

Wojskowa Akademia Techniczna

Tablice statyczne

- Tablice są podstawową złożoną strukturą danych
- Są uporządkowanymi agregatami danych tego samego typu o wspólnym identyfikatorze (nazwie)
- Do poszczególnych elementów mamy dostęp poprzez ich numer kolejny w tablicy
- Rozmiar tablicy statycznej jest znany w momencie kompilacji
- Numerowanie elementów zawsze rozpoczyna się od zera
- Jeśli tab jest tablicą, to jej pierwszy element to tab[0],a element o indeksie k jest oznaczany tab[k]

```
const int N = 20;
      int tab1[100],
          tab2[N],
          tab3[] = {1,2,3,4,5},
          tab4[5] = {1,2};
```

Zad. 3. Napisz program, który wpisze wszystkie elementy zdefiniowanych powyżej tablic.

Wojskowa Akademia Techniczna

Tablice, typ tablicowy

- Tablice tworzone na stosie jako zmienne lokalne muszą być deklarowane z rozmiarem, który jest znany w czasie kompilacji.
- Tam gdzie widoczna jest deklaracja tablicy tab, nazwa tab oznacza zmienną typu tablicowego o określonym rozmiarze.
- Na przykład po int tab[100]; typem tab jest int[100], a więc *stuelementowa tablica liczb całkowitych typu* int. Wymiar jest elementem specyfikacji typu: typy int[6] i int[5] są *różnymi typami*.

```
int tab [100]
```

Zad. 4. Napisz program, który sprawdzi prawdziwość ostatniego podpunktu.
Możesz wykorzystać:
sizeof(tab) / **sizeof**(**int**)

Wojskowa Akademia Techniczna

Tablica czy wskaźnik

- W prawie każdej operacji wykonywanej na zmiennej tab zmienna ta jest niejawnie konwertowana do typu wskaźnikowego.
- Wartością tab jest wtedy *adres* pierwszego elementu tablicy, a więc elementu o indeksie zero.
- Po deklaracji **int** tab[20]; zmienna tab może być traktowana jako wskaźnik typu **int* const** wskazujący na pierwszy element tablicy.
- Modyfikator **const** znaczy tutaj, że zawartość tablicy może być zmieniana, ale nie można do tab wpisać adresu innej tablicy.
- Ponieważ tab może być przekonwertowane do wskaźnika wskazującego na pierwszy element, więc wyrażenie *tab jest niczym innym jak nazwą pierwszego elementu tablicy. A &tab[0] jest równe wartości wyrażenia tab.

Zad. 5. Napisz program, który sprawdzi prawdziwość ostatniego podpunktu.

Wojskowa Akademia Techniczna

Arytmetyka wskaźników

- Do wskaźników można dodawać liczby całkowite.
- Typ wskaźnikowy *nie jest* jednak typem całkowitym, takie dodawanie jest zdefiniowane w specjalny sposób.
- Załóżmy, że p jest wskaźnikiem typu Typ*, a zmienna shift jest zmienną typu całkowitego (ale nie **long**).
- Wartością wyrażenia p+shift jest wtedy adres zawarty w zmiennej p powiększony o shift wielokrotności wymiaru zmiennej typu Typ (czyli **sizeof**(Typ)).
- Wyrażenie **p[i]** jest równoważne *(p+i).

Zad. 6. Sprawdzić przedostatni i ostatni punkt.
Użyj:
reinterpret_cast<long>
reinterpret_cast<long>(p+shift) == **reinterpret_cast<long>**(p) + shift * **sizeof**(p)

Zad. 7. Napisz program który wypiszę wszystkie elementy tablicy 8-elementowej korzystając z różnych sposobów odwołania się do poszczególnych elementów tablicy (co najmniej 3).

Wojskowa Akademia Techniczna

Tablice znaków

- Specjalne są tablice i wskaźniki znakowe. Związane jest to z faktem, że w klasycznym C nie ma typu napisowego, a rolę zmiennych napisowych pełnią tablice znaków
- Koniec napisu oznaczany jest znakiem ' \0'.
- Znak ten nazywany jest NUL; to znak o kodzie ASCII równym zeru który nie odpowiada żadnemu znakowi graficznemu
- (nie należy go mylić ze *wskaźnikiem* pustym NULL*)*.

Zad. 8. Napisz program który porówna rozmiar NUL oraz NULL.

Wojskowa Akademia Techniczna

Tablice znaków

```
int main() {
    char  tab1[] = "Kasia";
    char  tab2[] = {'B', 'a', 's', 'i', 'a', '\0'};
    const char *tab3 = "Wisia";
    cout << "Wymiar    tab1: "  << sizeof(tab1)   << endl;
    cout << "Wymiar    tab2: "  << sizeof(tab2)   << endl;
    cout << "Wymiar    tab3: "  << sizeof(tab3)   << endl;
    cout << "Wymiar \'Wisia\': "<< sizeof("Wisia")<< endl;

    tab1[0] = 'C';
    tab2[0] = 'C';
    tab3[0] = 'C';
    cout << "tab1: "  << tab1  << endl;
    cout << "tab2: "  << tab2  << endl;
    cout << "tab3: "  << tab3  << endl;

    return 0;}
```

Zad. 9. Co wypisze ten program?

Wojskowa Akademia Techniczna

Tablice wielowymiarowe

- Tablice w C/C++ mogą być wielowymiarowe, choć ich implementacja nie jest tak efektywna jak w innych językach programowania (w szczególności Fortranie).
- Tablica *n*-wymiarowa jest jednowymiarową tablicą wskaźników do tablic (*n* - 1)-wymiarowych
- Deklaracje (tablicy 2x4):

```
int tab[2][4];
```

```
const int dim1 = 2;
const int dim2 = 4;

int tab[dim1][dim2];
```

- Definicje:

```
int tab[2][4] = { 1,2,3,4,5,6,7,8 };
int tab[2][4] = { {1,2,3,4}, {5,6,7,8} };
```

Zad. 10. Napisz program, który wypełni macierz 3x4 elementami wprowadzanymi z klawaitury, a następnie je wypisze. Użyj pętli for.

Zad. 11. Napisz program, który w danej tablicy dwuwymiarowej zamieni miejscami dwa wiersze zdefiniowane przez użytkownika.

Wojskowa Akademia Techniczna

Typ tablicowy 2.0.

```
int tab[2][4];
```

- Jeśli tab[i] jest równoważne *(tab+i), to analogicznie tab[i][j], to to samo co *(tab[i]+j).
- Zatem tab[i] musi być typu wskaźnikowego – w tym przypadku **int***.
- tab[i] *wskazuje* na początek wiersza o indeksie i.
- tab[i] to *(tab+i) i wyłuskana wartość ma być typu **int***, a więc tab musi być typu „wskaźnik do wskaźnika do **int**”.
- Zatem taki typ tablicowy odpowiada typowi **int****.
- Odpowiada, ale nie jest z nim tożsamy!

Zad. 12. Jak będzie wyglądać odwołanie się do 2 kolumny, 3 wiersza przy pomocy:

- []
- *

w tablicy dwuwymiarowej?

```
tab[3-1][2-1]          =>    tab[2][1]
*(*(tab+3-1)+2-1)      =>    *(*(tab+2)+1)
```

Wojskowa Akademia Techniczna

Tablice napisów

- Tablica napisów to dwuwymiarowa tablica znaków, np:

```
char tab[2][70];
```

- Przeanalizuj kod programu….

```
int main() {
    const char **v;
    const char *t[] = {"abcd", "efghi", "jklmno" };
    v = t;
    cout << "v+2          = " << v+2          << endl;
    cout << "v[2]         = " << v[2]         << endl;
    cout << "*(v+2)       = " << *(v+2)       << endl;

    cout << "*(*(t+1)+2)  = " << *(*(t+1)+2)  << endl;
    cout << "t[1][2]      = " << t[1][2]      << endl;

    cout << "*(*(v+1)+2)  = " << *(*(v+1)+2)  << endl;
    cout << "v[1][2]      = " << v[1][2]      << endl;
}
```

```
v+2          = 0x7fff364d7080
v[2]         = jklmno
*(v+2)       = jklmno

*(*(t+1)+2)  = g
t[1][2]      = g

*(*(v+1)+2)  = g
v[1][2]      = g
```

Zad. 13. … i napisz program, który wypełni macierz 3x4 elementami wprowadzanymi z klawaitury, a następnie je wypisze. Używaj arytmetyki wskaźnikowej.