Programowanie komputerowe

Zajęcia 4

Jak przechowywane są dane?

- Do tej pory pokazaliśmy tworzenie zmiennych i przekazywanie ich jako argumentów do funkcji
- Każda zmienna, na którą można coś przypisać (znajduje się po lewej stronie znaku przypisania) ma swój **adres** w pamięci komputera
- Dlatego np. `int i=2` jest prawidłowe (zmienna `i` otrzymała adres w pamięci komputera, można na nią przypisać), ale `2=i` już nie (`2` nie jest zmienną, nie dostaje adresu, do którego można przypisać)
- Będziemy pracować z dwoma operacjami:
  - Znajdowaniem adresu do danej zmiennej
  - Sprawdzaniem, jaka wartość znajduje się pod danym adresem

Referencje

Jeśli `T` jest pewnym typem, to `T&` jest typem referencyjnym do `T`. Wartością typu referencyjnego jest **zmienna** typu `T`.

Typu referencyjnego możemy użyć jako parametru funkcji. Wówczas nie powstaje kopia wartości, a funkcja wywoływana działa bezpośrednio na zmiennej przekazanej jej jako parametr (dostała się do jej adresu). Robimy tak jeśli:

- Chcemy, żeby efekt funkcji na zmiennych przekazanych jako parametr pozostał nawet po zakończeniu działania funkcji,
- Nie chcemy (lub nie możemy) tworzyć kopii istniejącej zmiennej.

Przekazanie parametru za pomocą referencji nazywamy przekazaniem **przez zmienną**, a w zwykły sposób przekazaniem **przez wartość**.

Referencje – przykład

Poniższa funkcja zamienia wartości dwóch zmiennych:

```
void zamien(int& a, int& b) {
    int c=a;
    a=b;
    b=c;
}
```

Przykład użycia poniżej. Co się stanie jeśli pominiemy znaczki & ?

```
int main() {
    int x=3, y=7;
    zamien(x,y);
    cout << "x=" << x << ", y=" << y << endl;
}
```

Wskaźniki

- Wskaźnik to zmienna, której wartością jest miejsce w pamięci komputera, w którym przechowywana jest wartość pewnego typu (adres)
- Typ wskaźnikowy do typu `T` to `T*`
- Wartością wskaźnika może być `nullptr`, co oznacza, że wskaźnik na nic nie wskazuje
- **Wartość** wskazywaną przez wskaźnik `p` otrzymujemy przez `*p`
- Wskaźnik do zmiennej `v` uzyskujemy przez `&v`.

Operacje na wskaźnikach

- Wskaźniki można przesuwać operatorami `++`,`--`, `+`, `-`.
- `++` (`--`) przesunie wskaźnik o jedno miejsce w prawo (lewo).
- Dodanie (odjęcie) liczby całkowitej powoduje przesunięcie wskaźnika o podaną liczbę miejsc w prawo (lewo).

Podsumowanie pracy z adresami

- Znajdujemy się w budynku, który mieści nasz Wydział. Ten budynek nie zawsze był WNE UW – możemy potraktować go jako zmienną, która akurat przyjęła wartość WNE UW
- Budynek ma swój adres (Długa 40/44)
- Operator * – *co znajduje się pod adresem Długa 40/44?*
  - Obecnie odpowiemy, że WNE UW
- Operator & – *jaki jest adres WNE UW?*
  - Odpowiemy, że Długa 40/44 a np. ekipa remontowa będzie mogła wejść z pracami budowlanymi

Tablice

- Tablice służą do (uporządkowanego) przechowywania wielu wartości tego samego typu.
- Tablicę deklaruje się następująco:

```
typ nazwa[liczba_elementów];
```

- Można ustawić elementy tablicy w momencie deklaracji, np. tak:

```
int t[5] = {3, 7, -1, 2, 4};
```

- Inicjalizator nie musi mieć podanych wartości, ale wielkości tablicy potrzebuje

```
int tab[10] = {};
```

Tablice – dostęp do elementów

- Poszczególne elementy tablicy zachowują się jak zwykłe zmienne
- Elementy tablicy są indeksowane: pierwszy element znajduje się pod indeksem zero
- Żeby się dostać do elementu tablicy o konkretnym indeksie używamy nazwy tablicy i numeru wpisanego w nawiasy kwadratowe, np.

  `t[0]` – pierwszy element tablicy `t`

  `tab[10]` – jedenasty element tablicy `tab`

Tablice jako parametry funkcji

- Tablice są zawsze przekazywane przez zmienną – funkcja nie działa na kopii tylko na oryginalnej tablicy.
- Nie jest przekazywany rozmiar tablicy; żeby go przekazać należy użyć dodatkowego parametru, np.

```
int f(int tablica[], int rozmiar) { … }
```

- Przykład funkcji przyjmującej tablicę jako parametr:

```
void drukuj_tablice(int tab[], int n) {
    for (int i=0; i<n; ++i)
        cout <<tab[i] <<" ";
    cout <<"\n";
}
```

Ćwiczenia

Napisać poniższe funkcje. Parametr `r` oznacza długość tablicy `t`.

1. `void wczytaj(int t[], int r)`
   która prosi o podanie wartości dla tablicy `t`.
2. `int max(int t[], int r)`
   która zwraca największy element tablicy `t`.
3. `int suma(int t[], int r)`
   która zwraca sumę elementów tablicy `t`.
4. `bool rowne(int t1[], int t2[], int r)`
   która zwraca true jeśli tablice `t1` i `t2` są równe (mają takie same elementy).

Ćwiczenia (2)

5. `int max2(int t[], int r)`
która zwraca drugi największy element tablicy `t`.

6. `double srednia(double t[], int n)`
zwracającą średnią elementów tablicy.

7. `double odchStd(double t[], int r)`
zwracającą odchylenie standardowe elementów tablicy `t`

8. `int nwd(int t[], int r)`
która zwraca największy wspólny dzielnik elementów tablicy `t`.

Ćwiczenia (3)

9. `bool niemalejaca(int t[], int r)`
która zwraca true jeśli `t[0]<=t[1]<= ... <=t[r-1]`.

10. `void naKoniec(int t[], int r)`
która przestawia pierwszy element tablicy `t` na jej koniec.

11. `void wspak(int t[], int r)`
która zamienia kolejność elementów tablicy na odwrotną (np. `[1 6 3 2]` zamienia na `[2 3 6 1]`).