Podstawy programowania.

Wykład 5

Pętle. Tablice.

Tablice

- Tablica jest strukturą danych, w której zbiór elementów tego samego typu jest przechowywany w kolejno następujących po sobie komórkach pamięci
- Dostęp do elementu tablicy uzyskuje się przez indeks elementu
  - w C/C++ indeksem pierwszego elementu jest 0
- Tablice mogą przechowywać elementy rozmaitych typów, także złożonych (omawianych później)
- Podstawowe definicje (ew. z inicjowaniem zerami):
  - int tab_int[123] = {0}; double tab_double[12]; char tab_char[16];
- Tablice jako struktura danych są powiązane z pętlami jako strukturami sterowania

Pętle

- Pętla jest konstrukcją sterującą stosowaną w celu wielokrotnego wykonania tego samego zestawu instrukcji
  - jednokrotne wykonanie zestawu instrukcji stanowi pojedynczą iterację pętli
  - formalnie, zestaw instrukcji to pojedyncza instrukcja lub blok instrukcji, tworzący instrukcje złożoną
- W pętlach podstawowymi elementami związanymi ze sterowaniem są:
  - jednorazowe nadanie stanu początkowego
    - oznacza to stan danych przed rozpoczęciem wykonania pętli
  - sprawdzanie warunku zakończenia pętli w każdej iteracji
    - warunek oznacza osiągnięcie pewnego zakładanego stanu
- W trakcie realizacji pętli dokonuje się modyfikacji danych
  - modyfikacja danych pozwala na przerwanie pętli w odpowiednim momencie (spełnienie warunku zakończenia pętli)
- Wykonanie pętli można ilustrować schematami blokowymi

Pętla for

- Ważnym rodzajem pętli jest pętla for, która może przybierać różne postacie
- W najbardziej podstawowym wariancie składnia pętli for określa podstawowe elementy sterowania wykonaniem pętli:

  for(int i=0; i<10; i++){ ... pojedyncza iteracja ... }

  - pętla jest powiązana z tzw. zmienną sterującą pętli
    - wykonanie pętli oznacza wykonanie określonej liczby iteracji
    - każda iteracja jest związana z pewną wartością zmiennej sterującej
  - składnia uwzględnia
    - nadanie wartości początkowej zmiennej sterującej
    - warunek, który ma spełniać zmienna sterująca, dla dalszej kontynuacji wykonania pętli
      - zaprzestanie spełniania warunku oznacza przerwanie wykonania peli
    - modyfikacje zmiennej sterującej **po** każdej iteracji

Pętla for

- Przykład:

```
#define LICZBA_WYDRUKOW 5
int i; // zmienna sterująca widoczna w całej funkcji main
// suma liczb - inicjowanie danych przed wejściem do pętli
int suma_liczb=0;
// for – w postaci "kanonicznej"
for(i = 1; i <= LICZBA_WYDRUKOW; i++){
   suma_liczb = suma_liczb + i; // suma_liczb += i;
   printf("Iteracja %d: suma liczb od 1 do %d wynosi %d\n",
            i, i, suma_liczb);
}
printf("Wartość zmiennej sterującej po wykonaniu pętli: i = %d\n", i);
```

- W przypadku standardowej pętli for możliwe jest określenie z góry liczby iteracji

Pętle

- Pętla while

  while(warunek){ ... iteracja pętli ... }

  - składnia wymaga tylko określenia warunku powtarzania iteracji
    - uwaga: każda liczba całkowita odpowiada wartości logicznej (0 – fałsz, każda inna wartość - prawda)
  - inicjowanie danych początkowych i modyfikacje danych w pętli określane są standardowym kodem
  - warunek sprawdzany jest **przed** wykonaniem pierwszej (i każdej kolejnej) iteracji
    - oznacza to, że możliwe jest niewykonanie żadnej iteracji
  - przykład:
    - i=0; while(i<5) { i++; }
- Pętla do

  do{ ... iteracja pętli ... } while(warunek);

  - pętla do różni się od pętli while tylko tym, że warunek jest sprawdzany **po** pierwszej (i każdej kolejnej iteracji)
    - oznacza to, że co najmniej jedna iteracja jest wykonana

Tablice i pętle – przykład

- Prosty problem: obliczanie sumy wyrazów w tablicy:
  - dane zapisane w: int tab_dane[12];

- Kod C/C++:

```
int tab_suma =0;
for(int i=0; i<12; i++) {
   tab_suma += tab_dane[i];
}
```

lub

```
int i=0; int tab_suma=0;
while(i<12){
   tab_suma += tab_dane[i];
   i++;
}
```

- Kod w asemblerze:

```
.L2
   cmp %ebx,  12     // 12 <> i ?
   jl .L4
   jmp .L3
.L4
   add (%ebx), %ecx // tab_suma += tab_dane[i]
   inc %ebx                // i++
   jmp .L2
.L3
```

Warunki kontynuacji lub przerwania pętli

- Każda z pętli (for , while, do) wykorzystuje warunek logiczny w celu zdecydowania o kontynuacji lub przerwaniu wykonywania iteracji
  - warunek jest wyrażeniem, które przyjmuje wartość prawdy (kontynuacja iteracji) lub fałszu (przerwanie iteracji)
    - każda liczba całkowita odpowiada wartości logicznej
      - 0 – fałsz, każda inna wartość – prawda
    - wyrażenie może być prostym porównaniem (i<5) lub złożonym wyrażeniem z szeregiem wykonywanych operacji:
      - i<zakres && (znak=getchar()) != EOF && znak != '\n'
      - n==tab[i++] (uwaga na rozróżnienie = i ==)
      - ++i < 5 (oraz wiele innych omawianych w dalszych wykładach)
      - każdorazowo należy dokładnie zanalizować efekt wyrażenia, w szczególności jego **wpływ na wartości występujących zmiennych,** oraz jego ostateczną wartość logiczną

Pętle

- Pętla for

  `for(w1; w2; w3){ ... iteracja pętli ... }`

  - jest równoważne:

    `w1; while( w2 ) { ....; w3; }`

    gdzie `w1`, `w2` i `w3` są dowolnymi wyrażeniami
  - dozwolona składnia pętli for jest bardzo elastyczna i pozwala na:
    - umieszczanie rozmaitych operacji (jednej lub wielu, oddzielonych operatorem przecinka) jako `w1` i `w3`
      - najczęściej są to przypisania:

        `for (i = 0, j = n; i < j; i++, j--) { ... }`
    - umieszczanie złożonych wyrażeń jako warunku `w2`

    `for (i=0; i<zakres && (znak=getchar()) != EOF && znak != '\n'; i++){ ... }`
    - pomijanie składowych definicji
      - `for(;;){...}` - pętla nieskończona

Instrukcje *break* i *continue*

- Instrukcja *break* służy do przerwania wykonania:
  - ciągu instrukcji w ramach instrukcji *switch*
  - ciągu instrukcji w ramach dowolnej pętli *(while, for, do)*

  i przeniesienia sterowania poza pętlę lub instrukcję *switch*

```
for (i = 0; i < n; i++) {
    if (a[i] == szukana_wartosc){
      szukana_pozycja = i;
      break; // przerwane wykonanie najbardziej wewnętrznej pętli
} }
```

- Instrukcja *continue* służy do przerwania ciagu instrukcji w ramach pojedynczej iteracji dowolnej pętli *(while, for, do)* i przeniesienia sterowania na koniec iteracji

```
for (i = 0; i < n; i++) {
    if (a[i] < 0)  continue; // pomijamy elementy ujemne
    ... // skomplikowane przetwarzanie elementów dodatnich
}
```

Instrukcja *goto*

- Instrukcja *goto* jest pozostałością dawnego stylu programowania, odpowiednikiem rozkazu skoku w językach asemblera procesorów
  - składnia: goto etykieta;
  - gdzie etykieta oznacza konkretne miejsce w programie zdefiniowane jako etykieta:
    - zasięgiem nazwy etykiety jest funkcja, w której jest zdefiniowana
    - najczęściej definicja etykiety jest osobną linią
- Jedynym sensownym użyciem *goto* jest wyjście na zewnątrz wielokrotnie zagnieżdżonych pętli
  - (kiedy nie wystarcza zastosowanie *break*):

```
for ( ...) {
    for ( ...) {
        if (blad) goto obsluga_bledu;
}   } // co najmniej dwie pętle dla uzasadnienia użycia goto
obsluga_bledu:
... // kod obsługi błędu
```

Pętla *while*

- Przykład pętli *while* w programie kopiowania standardowego wejścia (*stdin*) na standardowe wyjście (*stdout*):

```
#include <stdio.h> // definicje symboli związanych z funkcjami wejścia/wyjścia
                   // (w tym stdin i stdout oraz EOF (end of file) - liczbowa
                   // wartość reprezentująca koniec strumienia danych;
                   // plik jest także strumieniem danych )
void main( void )
{
   int c = getchar(); // int ponieważ EOF niekoniecznie jest znakiem
   while (c != EOF) {
      putchar(c); // liczba całkowita z odpowiedniego zakresu
                  // jest interpretowana jako znak
      c = getchar ();
   }
}
```

Pętla *do*

- Przykład pętli *do* w programie obliczania liczby PI

```
   // wzór: PI/4 = 1 - 1/3 + 1/5 - 1/7 + 1/9 itd. itp.
double suma=0.0;
int i=0;
do{
     int  j = 1 + 4*i;                    // i = 0, 1, 2,itd.,   j = 1, 5, 9, itd.
     suma += 1.0/j – 1.0/(j+2.0); // zawsze dodatnie
     i++;                                  // dla każdego i liczymy dwa wyrazy z sumy

// użycie instrukcji break dla zbyt wielu wyrazów w sumie
// można w miejscu .... dodać dodatkowe instrukcje, wydruki itd. itp.
     if( i==ceil(max_liczba_wyrazow/2.0) ) { .....; break; }

// warianty zakończenia : 1. oba warunki (brak dodatkowych informacji)
// } while(i<ceil(max_liczba_wyrazow/2.0) && 4.0/j-4.0/(j+2) > dokladnosc);

} while( ( 4.0/j - 4.0/(j+2.0) ) > dokladnosc ); // wariant zakończenia 2
double obliczone_PI = 4*suma;
```