Podstawy języka C – cz. III

dr inż. Maciej Kusy

Katedra Podstaw Elektroniki
Wydział Elektrotechniki i Informatyki
Politechnika Rzeszowska

Elektronika i Telekomunikacja, sem. 2

Plan wykładu

- Typy pochodne: wskaźniki i tablice
- Dynamiczny przydział pamięci dla łańcuchów znakowych
- Typy złożone (struktury, unie)
- Strumienie w pracy z plikami
- Etapy pracy z plikiem
- Zapis i odczyt – wybrane funkcje biblioteki **`stdio`**

Typy pochodne: wskaźniki i tablice

Wskaźnik można użyć jako tablicę o dynamicznie ustalonej długości.

Takiemu wskaźnikowi należy najpierw przydzielić pamięć na zadaną liczbę elementów:

```
int n;    // n > 0
double* tab;
printf("Podaj liczbe elementow: ");
scanf("%i", &n);
tab = (double*)malloc(sizeof(double) * n);
tab[0] = 3.14; // *(tab+0) = 3.14;
printf("%f\n", tab[0]);
```

Przydzieloną pamięć należy następnie zwolnić: `free(tab);`

Dynamiczny przydział pamięci dla łańcuchów znakowych

Wskaźnik można użyć do przechowywania łańcuchów znakowych (stringów) o dowolnej długości.

```
char tablica[20];
char* nazwisko;
printf("Podaj nazwisko: ");
scanf("%s", tablica);
nazwisko = malloc(strlen(tablica) + 1);
strcpy(nazwisko, tablica);
printf("nazwisko: %s\n", nazwisko);
free(nazwisko);
```

dynamicznie obliczona długość

przekopiowanie do zmiennej **nazwisko**

Typy złożone – struktura

Struktura – typ, który grupuje w sobie dane różnego typu.

Strukturę definiuje się za pomocą słowa kluczowego **`struct`** i podanie jej nazwy:

```
struct nazwa
{
    //deklaracje pól różnych typów
};
```

Dane w strukturze grupowane są w jednym obszarze pamięci.

Struktura – przykład

Definicja struktury: słowo kluczowe **`struct`**, nazwa i umieszczone w nawiasach klamrowych deklaracje wszystkich pól składowych (średnik po klamrze zamykającej), np.:

```
struct Wydzial
{
      char nazwa[20];
      int liczba_Studentow;
};

struct Wydzial WEiI;
WEiI.liczba_Studentow = 177;
```

stworzenie obiektu struktury

odniesienie się do pola

Właściwości struktur

- Struktura jest typem, który grupuje w sobie dane różnego typu w jednym obszarze pamięci;
- Składowymi struktur są pola posiadające określone nazwy;
- Składowymi struktur mogą być zmienne typów prostych (całkowitych, zmiennoprzecinkowych) oraz typy pochodne (tablice, wskaźniki);
- Składową struktury może być również obiekt innej (wcześniej zdefiniowanej) struktury;
- Dostęp do wszystkich składowych struktury jest publiczny;
- Obiekty struktur można przesyłać do funkcji poprzez wartość lub za pomocą wskaźnika;
- Rozmiar struktury (w bajtach) może być większy niż suma rozmiarów poszczególnych pól struktury.

Typy złożone – unia

Unia – typ podobny do struktury. Wszystkie pola zajmują ten sam obszar pamięci. Rozmiar unii równy jest rozmiarowi największego składnika.

```
union nazwa
{
     //deklaracje pól różnych typów
};
```

Użycie unii ma na celu zmniejszenie zapotrzebowania na pamięć (wykorzystanie tylko jednego pola); często łączone jest z użyciem struktur.

Strumienie w pracy z plikami

- Strumienie obejmują: źródła odczytu (klawiatura i plik) oraz źródła zapisu (ekran i plik);
- Każdy program w momencie uruchomienia posiada od razu trzy otwarte strumienie:
  - **`stdin`**: pobieranie informacji ze strumienia (np.: wczytywanie z klawiatury, odczyt z pliku),
  - **`stdout`**: wstawianie informacji do strumienia (np.: wypisywanie na ekran, zapis do pliku),
  - **`stderr`**: powiadamianie o błędach;
- Aby korzystać ze strumieni należy dołączyć bibliotekę **`stdio.h`**;
- W celu skojarzenia pliku ze strumieniem należy stworzyć wskaźnik na strukturę typu **`FILE`** (przechowuje ona dane o pliku, np. aktualna pozycja w pliku);
- Wczytanie całkowitej informacji z pliku umożliwia stała **`EOF`** (*ang. **E**nd **O**f **F**ile*), która reprezentuje jego koniec.

Etapy pracy z plikiem

- Stworzenie wskaźnika na strukturę typu **`FILE`**.
- Zainicjalizowanie tego wskaźnika poprzez wywołanie funkcji o sygnaturze:

```
FILE * fopen(const char * filename,
             const char * mode);
```

gdzie:
– **`filename`**: nazwa pliku;
– **`mode`**: tryb dostępu do pliku:
`r` - odczyt,
`w` - zapis,
`a` - dołączanie zawartości do istniejącego pliku – jeżeli plik nie istnieje, funkcja tworzy nowy plik,
`r+`, **`w+`**, **`a+`**.

- Dokonanie operacji zapisu / odczytu.
- Zamknięcie pliku poprzez wywołanie funkcji o sygnaturze:

```
int fclose(FILE * fp);
```

Zapis – wybrane funkcje **`stdio.h`**

Zapisanie pojedynczego znaku (**`c`**) do pliku:

```
int fputc(int c, FILE * fp);
```

Argument **`c`** konwertowany jest do **`unsigned char`**.
Funkcja zwraca zapisany znak lub **`EOF`** w przypadku błędu.

Zapisanie łańcucha (**`s`**) do pliku:

```
int fputs(const char *s, FILE * fp);
```

Funkcja zwraca wartość niezerową przy poprawnym zapisie lub **`EOF`** w przypadku błędu.

Zapisanie łańcucha do pliku:

```
int fprintf(FILE *fp,
            const char * format, ...);
```

Odczyt – wybrane funkcje **stdio.h**

Odczytanie pojedynczego znaku z pliku:

```
int fgetc(FILE * fp);
```

Funkcja zwraca odczytany znak (kod ASCII) lub **EOF** w przypadku błędu.

Odczytanie **n-1** znaków z pliku:

```
char * fgets(char * buf, int n, FILE * fp);
```

Odczytany łańcuch funkcja kopiuje do zmiennej **buf** i wstawia **null** na koniec stringu.

Odczytanie łańcucha z pliku:

```
int fscanf(FILE *fp,
                const char * format, ...);
```

Funkcja przerywa czytanie po napotkaniu białego znaku.