**ЛАБОРАТОРНА РОБОТА № 8.**

**Структури та об’єднання в мові програмування С++**

**8.1. Мета роботи**

Навчитися використовувати структури та об’єднання у мові С++.

**8.2. Теоретичні відомості**

8.2.1. Структури

Оголошення структур

Одним з найбільш важливих кроків при проектуванні програми є вибір відповідного способу представлення даних. У багатьох випадках простих змінних або навіть масиву виявляється недостатньо. Мова С++ дозволяє розширити можливості представлення даних за допомогою змінних складних типів, тобто *змінних типу структура*. У своїй базовій формі струкри С++ є досить гнучким засобом, щоб представляти широке розмаїття даних, а також вона дає змогу винаходити нові форми. *Структура* – це набір з однієї або більше змінних, можливо різних типів, згрупованих під одним ім'ям для зручності обробки. Оголошення структури є генеральним планом, який описує спосіб формування структури. Іноді оголошення структури називають *шаблоном*:

**struct <ідентифікатор>**

**{**

**<тип> <ідентифікатор поля1>;**

**<тип> <ідентифікатор поля2>;**

**...**

**};**

Поняття структура застосовується в двох сенсах. Одним з них є шаблон, що повідомляє компілятору, як представляти дані, але не призводить до виділення простору в пам'яті для цих даних. Наступний крок полягає в створенні змінних типу структури, і в цьому полягає другий сенс поняття. Рядок програми, який створює змінну типу структури, має вигляд:

**struct <назва структури> <змінна1>, <змінна2>, ...;** Змінні можна також оголошувати при створенні шаблону структури, після закритої фігурної дужки.

Розглянемо приклад:

**struct DATE {**

**int day;**

**int year;**

**char monName[4];**

**} d;**

Опис структури, що складається з взятого в фігурні дужки списку полів, починається з ключового слова **struct**. Після слова **struct** пишеться ідентифікатор, так звана, *мітка* структури (тут це **DATE**). Ідентифікатор може бути відсутнім. Така мітка іменує структури цього виду і може використовуватися надалі як скорочений запис шаблону. Елементи або змінні, згадані в структурі, називаються *полями* (елементами, компонентами). Область видимості ідентифікаторів полів є в межах блоку шаблону структури. Оголошення **struct DATE d;** визначає змінну **d** як структуру типу **DATE**.

Структура не може містити в собі змінні свого типу, тобто наступне оголошення шаблону буде неправильним:

**struct man**

**{**

**char \*fam;**

**char \*name;**

**struct man any; //Error!**

**};**

Однак структура може включати елементи, що є вказівниками на оголошений тип цієї ж структури:

**struct man**

**{**

**char \*fam;**

**char \*name;**

**struct man \*pMan; //Ok**

**};**

Доступ до полів структури

Операція членства структури “**.**” зв'язує ім'я структури та ім'я члена. Вона називається *селектором*:

**<змінна типу структура>.<поле відповідної структури>;**

Наприклад:

**d.day**  **= 1;**

**d.year**  **= 2017;**

**d.monName**  **= "Jan";**

В програмі це може бути реалізовано наступним чином:

**#include <stdio.h>**

**#include <string.h> s**

**truct DATE**

**{ int day;**

**int year;**

**char monName[4];**

**};**

|  |
| --- |
| **int main(void)**  **{**  **struct DATE d;**  **d.day = 1;**  **d.year = 2017;**  **strcpy(d.monName, "Jan");**  **printf("Date - %d %s %d\n",**  **d.day, d.monName, d.year);**  **return 0;**  **}** |

Результатом виконання програми буде напис : **Date - 1 Jan 2017**.

Структури можуть бути вкладеними, наприклад:

**struct STUDENT**

**{**

**char**

**name[25];**

**char address[40];**

**long int zipCode;**

**long int stNumber;**

**double scholarship;**

**int course;**

**char group[7];**

**struct DATE birthDate;**

**struct DATE hereDate;**

**};**

Структура **STUDENT** містить дві структури типу **DATE** . Якщо змінна **person** визначається як **struct STUDENT person**, то запис:

**person.birthdate.year**

буде посилатися на рік народження студента.

Доступ до полів структури через вказівник

Над змінною структури можна виконати операції взяття адреси змінної та присвоєння її вказівнику:

**struct DATE d, \*p;**

**p = &d;**

Для демонстрації прикладу використання цих операцій над структурами перепишемо наведену вище програму так, щоб ввід даних про дату відбувався в фунrції **inputDate()***,* а вивід в процедурі **printDate()***.* В такому випадку програма набире наступного вигляду:

**#include <stdio.h>**

**#include <string.h> struct DATE**

**{**

|  |
| --- |
| **int day;**  **int year;**  **char monName[4];**  **};**  **struct DATE inputDate(**  **int day,**  **int year,**  **char \*month)**  **{**  **struct DATE tmp;**  **tmp.day = day;**  **tmp.year = year;**  **strcpy(tmp.monName, month);**  **return tmp;**  **}**  **void printDate(struct DATE \*d)**  **{ printf("Date - %d %s %d\n",**  **d->day, d->monName, d->year);**  **}**  **int main(void)**  **{**  **printDate(&inputDate(1, 2017,"Jan"));**  **return 0;**  **}** |

Розглянемо детальніше функцію **void printDate(struct DATE \* d)**. Як видно з оголошення функції, у якості формального параметра використано вказівник на структуру. В такому випадку для звертання до полів структури використовується селектор вказівника“**->**”:

**<вказівник типу структура> -> <поле структури>;**

**(\*<вказівник типу структура>).<поле структури>;**

Наприклад:

|  |  |
| --- | --- |
| **struct DATE d;**  **d.date = 1;**  **d.year = 2017; strcpy(d.monName, "Jan");** | **struct DATE d,\*d1;**  **d1 = &d;**  **d1->date = 1;**  **d1->year = 2017; strcpy(d1->monName,"Jan");** |

Масиви структур

Структури можна об'єднувати в масиви. Тоді запис **struct DATE d[5];** відображає масив з п’яти елементів типу **DATE**. Доступ до полів структури відбувається наступним чином:

**d[0].year; d[1].date;**

Розмір структури можна взнати використавши стандартну операцію. У мові С передбачена унарна операція **sizeof()**, яка виконується під час компіляції, дозволяючи обчислити розмір будь-якого об’єкта. Вираз:

**sizeof(object);**

повертає ціле число, що дорівнює розміру вказаного об’єкту.Об’єкт може бути фактичною змінною, масивом і структурою, або іменем основного типу, як **int** або **double**, або іменем похідного типу, як структура. Наприклад:

**int size;**

**struct DATE d;**

**size = sizeof(d);**

Ініціалізація структур

При визначенні структурних змінних можна ініціалізувати їхні поля. Ця можливість подібна ініціалізації масиву й слідує тим же правилам:

**<ім'я шаблону> <ім'я змінної структури> = {**

**<значення1>, <значення2>, ...**

**};**

Компілятор присвоює **<значення1>** першій змінній в структурі, **<значення2>** – другій змінній й т.д. Тут необхідно додержуватися деяких правил:

* значення, що присвоюються, повинні співпадати по типу з відповідними полями структури;
* можна оголошувати кількість значень, що присвоюються, меншу, ніж кілкість полів. Компілятор присвоїть нулі іншим полям структури;
* список ініціалізації послідовно присвоює значення полям структури, вкладених структур і масивів.

Наприклад:

**struct date { int day,month, year;}d[5] = {**

**{1, 3, 1980},**

**{5, 1, 1990},**

**{1, 1, 1983}};**

Проініціалізовані перші три елементи массиву.

8.2.2. Об'єднання

*Об’єднання* подібне структурі, але в кожен момент часу може використовуватись (або є активним) тільки один з його компонентів. Тип об’єднання може задаватися записом виду:

|  |
| --- |
| **union <МІТКА>**  **{**  **<тип> <компонент1>;**  **<тип> <компонент1>;**  **...**  **};** |

Для кожного з цих компонентів виділяється одна і та ж область пам’яті, тобто вони перекриваються. Хоча доступ до цієї області пам’яті можливий через використання будь-якого з компонентів. Компонент для цієї мети повинен вибиратися так, щоб отриманий результат не був беззмістовним. Доступ до компонент об’єднання відбувається тим самим способом, що і для структур.

Об’єднання застосовуються:

* для мінімізації об’єму пам’яті, що використовується, якщо в кожний момент часу тільки один об’єкт з багатьох є активним;
* для інтерпретації основного представлення об’єкта одного типу, так ніби цьому об’єкту був присвоєний інший тип.

В якості прикладу визначення об’єкта типу **union** розглянемо об’єднання **shape**, яке визначається наступним чином:

|  |
| --- |
| **struct position**  **{**  **int x; //sizeof(int) bytes**  **int y; //sizeof(int) bytes**  **};**  **union {**  **float radius; //circle sizeof(float) bytes**  **float a[2]; //rectangle sizeof(float)\*2 bytes int b[3]; //triangle sizeof(float)\*3 bytes struct position p;//coordinate sizeof(int)\*2 bytes } shape;** |

Розмір об’єднання **shape** дорівнює не сумі всіх байтів, як у випадку структури, а розміру найбільшого поля (компоненти). В цьому прикладі є сенс використовувати тільки той компонент, який отримав останнім своє значення.

С++ дає можливість використовувати у якості полів анонімні об’єднання та структури. *Анонімне об'єднання* (*або структура*) – це неіменоване об'єднання (або структура), що є членом структури або об'єднання.

Наприклад:

|  |
| --- |
| **struct names**  **{**  **char first[20];**  **char last[20];**  **};**  **struct person**  **{**  **int id;**  **struct names name;**  **};**  **struct person ted = {8483, {"Ted", "Grass"}};** |

Стандарт дає змогу визначати структуру **person**, використовуючи в якості члена вкладену неіменовану структуру:

**struct person**

**{**

**int id;**

**struct{char first[20]; char last[20];};**

**};**

Цю структуру можна було б ініціювати в тій же манері:

**struct person ted = {8483, {"Ted", "Grass"}};**

Але доступ до членів спрощується, оскільки для цього застосовуються імена членів як **first**, так, ніби вони були членами **person**:

**puts(ted.first);**

Зрозуміло, можна було б просто зробити **first** і **last** безпосередніми членами структури **person**, позбувшись від вкладеної структури. Засіб анонімності є більш корисний з вкладеними об'єднаннями.

8.2.3. Вирівнювання

Кожен тип об'єкту має властивість, що називається *вирівнюванням*. Вирівнювання рівне цілому значенню типу **size\_t** (зазвичай **unsigned long**) та представляє число байтів між послідовними адресами, що можуть бути виділені для об'єкту даного типу. Згідно стандарту та апаратної реалізації оперативної пам’яті комп’ютерів, допустимі значення вирівнювання є цілими невід'ємними степенями двійки. Для того, щоб задовольнити вимоги до вирівнювання всіх членів структури, після деяких її членів компілятор може автоматично вставити відступи. Це обов’язково слід враховувати при використанні структур та об’єднань.

З появою стандарту С++11 з’явилися стандартні засоби мови С для контролю вирівнювання даних. Зокрема оператор часу компіляції:

**\_Alignof( type-name );**

Повертає вирівнювання типу **type-name**. Якщо **type-name** являє собою масив, результат є вирівнюванням елемента масиву. **type-name** не може бути функцією або неповним типом. Підключивши стандартну бібліотеку **<stdalign.h>**, цей оператор можна замінити на більш звичний **alignof()**.

Оператор часу компіляції:

**\_Alignas ( expression );**

**\_Alignas ( type );**

що може використовуватися тільки при оголошенні об'єктів. Він не може бути використаний в оголошеннях параметрів функції. При використанні в оголошеннях, заявлений об'єкт буде мати вирівнювання рівне:

* результату виразу, якщо він не дорівнює нулю і є додатнім степенем двійки;
* вирівнювання типу, тобто, до **alignof(type)** ;

Якщо вираз має значення нуль, то специфікатор не має ніякого ефекту. Підключивши стандартну бібліотеку **<stdalign.h>**, цей оператор можна замінити на більш звичний **alignas()**. Приклади:

|  |
| --- |
| **#include <stdalign.h> #include <stdio.h>**  **struct sse\_t**  **{**  **alignas(16) float sse\_data[4];**  **};**  **struct data**  **{**  **char x;**  **alignas(128) char cacheline[128];**  **};**  **int main(void)**  **{**  **printf("sizeof(data) = %ld (1 byte + 127\n"**  **"bytes padding + 128-byte array)\n",**  **sizeof(struct data));**  **printf("alignment of sse\_t is %ld\n",**  **alignof(struct sse\_t)); }** |

Результати виконання:

**sizeof(data) = 256 (1 byte + 127 bytes padding + 128-byte array) alignment of sse\_t is 16**

**8.3. Контрольні запитання**

1. Який загальний вигляд структури?
2. Як обчислити розмір структури?
3. Як доступитись до полів структури з змінної типу структура?
4. Як доступитись до полів структури з вказівником на змінну типу структура?
5. Який загальний вигляд об’єднання?
6. Яка принципова різниця між структурою та об’єднанням?
7. Що таке вирівнювання даних?

**8.4. Лабораторне завдання**

1. Навчитися використовувати структури та об’єднання у мові С++.
2. Одержати індивідуальне завдання.
3. Побудувати блок-схеми алгоритмів відповідно до завдання.
4. Скласти програми на алгоритмічній мові C++ згідно завдання.
5. Відлагодити програми, виконати обчислення, проаналізувати отримані результати.

**8.5. Зміст звіту**

1. Титульний аркуш.

1. Мета роботи.
2. Індивідуальне завдання.
3. Блок-схема алгоритмів у відповідності до завдання.
4. Тексти програм у відповідності до завдання.
5. Результати обчислень.
6. Аналіз результатів, висновки.

**8.6. Індивідуальні завдання**

Завдання 1

Скласти програму що дає змогу з використанням структур та об'єднань реалізувати розв’язок поставленої задачі. Всі вхідні дані беруться з текстового файлу (створити не менше десяти відповідних записів у файлі). Ввід-вивід даних та виконання інших окремих логічних дій необхідно реалізувати в окремих функціях. У головній функції необхідно виконувати лише їх виклик. Використання глобальних змінних не допускається. Інформація повинна передаватися у функції лише за допомогою параметрів.

1. В бібліографічному каталозі знайти книги по алгоритмічній мові С++. В каталог заноситься шифр книги, прізвище автора, назва, рік видання і кількість сторінок.
2. При обробці інформації про стан здоров’я студентів групи знайти найвищого і найважчого. В медичну картку входить інформація: прізвище, рік народження, ріст, вага.
3. В списку студентів групи обчислити кількість студентів з іменами, що користувач вводить з клавіатури.
4. Знайти найхолодніший і найтепліший день квітня. В метеорологічній інформації міститься: день місяця, температура, % вологості, опади (дощ, сніг тощо).
5. В файлі записана інформація про студентів груп бакалаврату, яка складається з прізвища, імені, статі і віку. Вивести на друк шифр групи в якій найбільший % дівчат.
6. В файлі записано результати екзаменаційного контролю студентів групи. Скласти програму, яка по кількості набраних балів буде визначати оцінку кожного студента. Рахувати що: 100-88 балів це 5; 87-71 балів – 4; 70-51 – 3; менше 50 – 2.
7. В файлі записано прізвище студента і його оцінки за останню сесію. Вивести на друк прізвище і середній бал студентів, як мають середній бал вищий за середній бал групи.
8. Передбачити запис нових абонентів у телефонний довідник сортуючи їх по номеру.
9. У відомості студентів групи записано: Прізвище і ініціали, адреса і телефон. В разі відсутності телефону в графі, що відповідає цій інформації стоять пробіли. Скласти програму яка буде друкувати номер телефону потрібного студента, а в разі відсутності телефону його адресу.
10. По результатах метеорологічних досліджень: дата, середньодобова температура, опади (дощ, сніг тощо), визначити найтепліший день місяця.
11. Передбачити запис нових абонентів у телефонний довідник, якщо вже записані абоненти розміщені там по алфавіту.
12. В каталозі записано назви книг в алфавітному порядку. В інформацію про книгу входять. Прізвище автора, назва книги, рік видання. Скласти програму яка дозволить доповнювати каталог новими назвами і буде розміщати їх а алфавітному порядку враховуючи і вже існуючі назви.
13. У вхідному файлі записано відомість на стипендію студентів групи, де входить прізвище, ім’я, по-батькові і розмір стипендії. Скласти програму що виведе інформацію про стипендію будь-якого заданого студента.
14. Визначити статистику оцінок, одержаних студентами групи (кількість п’ятірок, четвірок, трійок, двійок) під час сесії.
15. У вхідному файлі записана відомість, де входить прізвище, ініціали, кількість набраних балів. Вивести на друк розсортовану відомість по прізвіщу і по набраних балах та середній бал групи.
16. У вхідному файлі записано телефонний довідник скласти програму, яка по заданому прізвищу буде виводити номер телефону абонента.

Завдання 2

Розробити програму яку забезпечує опрацювання структур даних. Необхідно забезпечити опрацювання 3-5 полів елементів з використанням різних простих типів даних (стрічки, символи, числа). Забезпечити виконання таких операцій:

* + додавання нового елементу;
  + пошук елементу за значенням полів;
  + послідовний перегляд елементів;
  + модифікація значень полів елемену;
  + видалення елементу;
  + сортування за значеннями полів.

Результати всіх операцій повинні зберігатись у файлі (створити не менше десяти відповідних записів у файлі). Елемент:

1. Книга (назва, автор, видавництво, рік, кількість сторінок).
2. Студент (ПІБ, інститут, група, курс, рейтинговий бал).
3. Автомобіль (виробник, модель, ціна, потужність, рік випуску).
4. Мікросхема (марка, розрядність, обсяг, ціна).
5. Фрукт (назва, країна-постачальник, ціна).
6. Музика (назва, виконавець, альбом, жанр, рік).
7. Замовлення в бюро перекладів (клієнт, мова, к-сть. сторінок, перекладач, дата, вартість).
8. Фільм (назва, рік, рейтинг, режисер, жанр, бюджет).
9. Текстові редактори: (назва, виробник, ліцензія, рейтинг, вартість).
10. Відомості про сім'ю студента (ПІБ, інститут, професія батька, професія матері, кількість братів і сестер).
11. Міський транспорт (вид, номер, початкова зупинка, кінцева зупинка, кількість зупинок, час в дорозі).
12. Насіння (тип, назва рослини, вага насіння, ціна).
13. Побутова техніка (тип, назва, фірма виробника, ціна).
14. Країна (назва, кількість населення, площа).
15. Одяг (тип, назва, розмір, ціна).
16. Фарба (тип, колір, об’єм фасування, ціна).