**構造体**

　構造体とは、変数を一つにまとめて定義されるユーザー定義の型になります。

1. **ユーザー定義の型**

ユーザー定義の型とは構造体やクラスのことを言います。C言語やC++で扱える変数の型は大きく区別すると下記の二つがあります。

|  |
| --- |
| **組み込み型**  　int ,short, char, longなど  **ユーザー定義型**  　構造体、クラス、enumなど |

1. **なぜ構造体を使うのか？**

変数を意味のある一つの塊にまとめると可読性が上がり、コードの保守性も向上するため。

1. **構造体の書き方。**

では、ゲームでよくあるプレイヤーに関する変数を構造体に変更する方法を記述します。

変更前

|  |
| --- |
| int main(){  　　int playerHp = 100; //プレイヤーのHP  int playerMp = 200; //プレイヤーのMP  int playerLv = 20; //プレイヤーのレベル  std::cout << playerHp;  std::cout << playerMp;  std::cout << playerLv;  } |

　 変更後

|  |
| --- |
| //Playerという新しいユーザー型を定義。  //これが構造体。  struct Player{  　int hp; //HP  int mp; //MP  int lv; //レベル  };  int main()  {  //Player型の変数のplayerを定義。  //定義の仕方は組み込み型と同じ。  Player player;  std::cout << player.hp;  std::cout << player.mp;  std::cout << player.lv;    } |

**グローバル変数とローカル変数とスタティック変数**

C言語にはグローバル変数とローカル変数とスタティック変数というものがあります。

これらの変数の違いは何かというと、変数のスコープが異なります。

(スタティック変数について今回は説明しません。スタティック変数の説明は後日C++Ⅲで解説予定のファイル分割の時に説明します。)

1. **スコープ**

スコープとはその変数の有効範囲を意味します。例えば下記のplayerHpという変数を考えてみましょう。

|  |
| --- |
| void FuncA()  {  int playerHp; //プレイヤーのHPを定義。これはローカル変数。  playerHp = 100; //プレイヤーのHPに100を設定。  　 std::cout << playerHp;  }  void FuncB()  {  std::cout << playerHp; //コンパイルエラー！！！playerHpは定義されていない。！！  } |

playerHpは関数内で定義されているため、ローカル変数と言われる変数になります。

ローカル変数とは定義された関数ないでのみ有効なため、playerHpはFuncBで使用することはできず、playerHpは定義されていない！というコンパイルエラーが発生します。

ローカル変数だけではプレイヤーのHPのような、ゲーム中いたるところで使用しそうな変数の扱いに困ってしまいます。そのため、すべての関数でアクセス可能な変数、グローバル変数という変数が存在します。ではplayerHpをグローバル変数に変更してみましょう。

|  |
| --- |
| int playerHp; //プレイヤーのHPを定義。これはグローバル変数。  void FuncA()  {  playerHp = 100; //プレイヤーのHPに100を設定。  　 std::cout << playerHp; //プレイヤーのHPを表示。  }  void FuncB()  {  　 std::cout << playerHp; //グローバル変数なのでアクセスできる。  } |

このように、playerHpはグローバル変数になったため、プログラムのどこからでも参照できるようになりました。

また、ローカル変数は関数を抜けると破棄されてしまうため、関数を抜けた後は値を保持しておくことができません。ゲームプレイ中など永続的に値を保持していたい場合はグローバル変数を使用することになるでしょう。

|  |
| --- |
| *Tips*  *実はグローバル変数の乱用は決して褒められたことではありません。極端な話、ローカル変数を一切使わずに、すべてグローバル変数を使用してプログラムを書くことも可能です。しかし、そのようなコードを書いていた場合プログラムが巨大になってきたときに、あなたは必ず後悔します。そのため、シングルトンパターンなどのようにグローバル変数をもう少しマシに使えるようにするためのテクニックが存在します。*  *ただし、今はグローバル変数の扱いに慣れるため、積極的にグローバル変数を使用してかまいません。慣れたころにこのtipsの話を思い出してください。* |